An Introduction to Life-Cycle Models

Robert Kirkby

Victoria University of Wellington

https://doi.org/10.5281/zenodo.8136810

March 2, 2025

Abstract

Introduces life-cycle models and shows how to solve them and build more realistic versions. Covers most of the main concepts in 'one-asset' life-cycle models with consumption-leisure and consumption-savings decisions. Proceed by a series of models, with the intention that you can read the pdf description of a model, then look through the codes that show how to implement it. By the end you can easily solve one-asset finite-horizon problems, including plotting life-cycle profiles and simulating panel data. A few of the models are dedicated to describing key concepts of borrowing constraints, precautionary savings, and incomplete markets.

Keywords: Life-cycle model.

JEL Classification:

1 Introduction

We gradually build up a life-cycle model that can be used to look at income, hours worked, consumption, and assets over the life-cycle. We will start with a deterministic life-cycle model in which people live for J periods and make decisions on how much to work. Our second model will then add a decision about how much to save (assets). Our third model will just use this model to draw a life-cycle profile. We will then step-by-step make additions to the model to understand how these help us create more realistic life-cycle profiles including idiosyncratic shocks. This pdf explains the models one-by-one, and the codes implementing each of them are all on github. We make use of the VFI Toolkit (vfitoolkit.com). The intention is that you can go through the models one-by-one, first reading the pdf explanation of the model and then running the codes and seeing how to implement it.

By the end we will have a life-cycle model in which people make consumption-savings and consumption-leisure choices, which has working age and retirement, which is capable of capturing that earnings are hump-shaped over

age (peaking around ages 45-55), the variance of both income and consumption increase with age, incomes grow in line with deterministic economic growth of the economy as a whole, people have some assets left when they die, people face the risk of substantial medical costs when old, and where borrowing constraints and precautionary savings play an important role. And we will be able to use these to plot life-cycles, including mean, variance, and gini coefficients conditional on age, and even on 5 year age-bins. We will also be easily able to simulate panel data sets from the model on which we could run regressions.

These life-cycle models can be used easily requiring very little knowledge of numerical methods; all you need is Matlab and a gpu.[1] They are intended as a simple way to understand and use life-cycle models. Follow-up documents will then build on these to explain how to extend them to general equilibrium OLG (overlapping-generation) models, and how to solve general equilibrium transition paths for OLG models. The codes are an 'introduction' to life-cycle models, not the final word. If you find this interesting and want to start working with models that have two or three assets you will need to learn to write (much faster) code yourself; I recommend looking up the EGM (endogenous grid method) for solving finite-horizon models and starting from there.[2] Obviously from the perspective of the codes there is no need to stick to 'hours worked', 'assets', and 'labor productivity shocks' as the decision variable, endogenous state, and exogenous state, they can be anything you want to call them/interpret them as.

Note that the calibration of the model parameters has not be taken seriously in these example models. If you do any work with life-cycle models it is very important that you take very seriously the question of what the appropriate parameter values are (whether by calibration, method of moments estimation, or anything else). I have not done so as I wish to focus on the concepts around setting up and solving life-cycle models. Near the end, there is Section on estimating life-cycle models by the Generalized Method of Moments that helps you understand how to perform formal statistical estimation of these models.

I have put in enough grids points on assets to make the solutions reasonably accurate, but I recommend you use more to get full accuracy in practice, especially if looking at things like the tails of distributions, inequality, subpopulations, or correlations in panel data. I have erred on the low side while still having enough points to illustrate concepts simply so that the codes can be run on less powerful gpus and therefore be more widely accessible.

If you have any questions about the material, or spot a typo in the codes, or would just like to ask a clarifying question, etc., please use the forum: discourse.vfitoolkit.com

[1] The codes make automatic use of the gpu via the VFI Toolkit for Matlab; the gpu must be an NVIDIA gpu. I tested all of these codes on a gpu with 8gb of ram, vast majority of them would work with less gpu memory but no promises :)

[2] The VFI Toolkit just uses pure discretization (discretize all decision variables, next period endogenous state, this period endogenous state, and the exogenous states), this is simple and robust, but not nearly as good a combination of speed and accuracy and more sophisticated methods.

Contents

2 Building up Life-Cycle Models

2.1 Life-Cycle Model 1: Consumption-Leisure

Households live for J periods and solve the life-cycle problem,

$$\sum_{j=1}^{J} \beta^{j-1} \qquad \left[\frac{c_j^{1-\sigma}}{1-\sigma} - \psi\frac{h_j^{1+\eta}}{1+\eta}\right]$$
$$c_j = wh_j$$
$$0 \leq h_j \leq 1$$

where j indexes age, c_j is consumption, h_j is hours worked,[3] w is the wage per hour worked. So a household that works h_j hours receives an hourly wage of w, giving them a pre-tax labor income of wh_j. Notice that all the parameters are just a single number.[4]

We can rewrite this recursively as a value function problem,

$$\begin{aligned} V(j) \quad = \quad & \max_{c,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} + \beta V(j+1) \\ & c = wh \\ & 0 \leq h \leq 1, aprime \geq 0 \end{aligned}$$

notice that the value function V depends on the age j, so households face different problems and make different decisions (different optimal policies) at different ages.[5] There is one of these problems for each $j = 1, 2, ..., J$; one problem for each period of the agents life. We assume $V(J+1) = 0$.[6] Notice also from the budget constraint that once the household chooses one of c or h, it is implicitly choosing the other. The codes take advantage of this and just has one decision of h.

To solve this we need parameter values for β, σ, ϕ, η and w. In the codes we will create these. We also need to let the codes know what the 'return function' is: essentially this combines the period-utility function ($\frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta}$) with the constraints ($c = wh$ and $0 \leq h \leq 1$).[7] The only other thing the codes need is the 'grid' on h, and it needs to know which parameter is the discount factor —the parameter that discounts the next period

[3] h is actually 'fraction of time worked', but I will call it hours worked throughout simply as it is easier to think of in this way. $0 \leq h \leq 1$ is showing us that this 'fraction of time worked' is from 0 to 1.

[4] In principle we could write the budget constraint as $c_j \leq wh_j$, but it would anyway bind every period as otherwise the agent is essentially throwing income away by not consuming it all (there is nothing else to do with it in this model).

[5] While true of these models in general this is actually not true in this most simple example. In this first simple example the problems for each age are completely seperate as there is nothing connecting them, that is, there is no way (such as savings) to move resources between periods. Put another way, nothing the household does in one period can have any influence on any other period. As a result, for this simple example the household actually faces a seperate problem in each period, and because all the constraints, utility function and parameters are the same it is even the same problem at every age and so the household will make the same decisions at every age.

[6] $V(J+1)$ is effectively the utility received upon dying.

[7] In practice we will enforce $0 \leq h \leq 1$ via the grid on h, so only the budget constraint appears inside the code for the return function.

value function $V(j+1)$— which in our case is β.

We will set this up in the code and then use the command 'ValueFnIter_FHorz_Case1()' to solve to get both the value function and the policy function. The value function tells us the value at each point on the grid (currently the only variable the value function depends on is j). It also gives us the 'policy function', which is the optimal decision the agent makes, in our case this will just be the choice of h (by default the policy is the choice of grid point for h, rather than the value of h, the codes will make this clearer).

2.2 Life-Cycle Model 2: Retirement

Let's make just about the simplest change we can. Currently households live J periods, specifically they live 81 periods intended to represent ages 20 to 100 years old, inclusive. In reality we know many people retire at age 65 and receive a pension. We will add this to our codes. We need to tell the codes how to detect retirement (which we will make automatic at age 65) and how much the pension is. Let $Jr = 46$ be the retirement age (46=65-19, 19 is years before the first model period of age 20 years old). The life-cycle value function of the household is now,

$$\begin{aligned} V(j) \quad = \quad & \max_{c,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + \beta V(j+1) \\ & \text{if } j < Jr: \ c = wh \\ & \text{if } j >= Jr: \ c = pension \\ & 0 \leq h \leq 1, aprime \geq 0 \end{aligned}$$

notice that $j < Jr$ is pre-retirement (working age), and $j >= Jr$ is retirement during which the household can no longer earn money by working, but does receive a pension. Notice that the codes will need to know age j, which in the code we will call $agej$.

In terms of the codes this will mean creating the parameters (Jr, $agej$ and $pension$) and passing them into the return function.

2.3 Life-Cycle Model 3: Assets

We will now add assets to the household problem. This will give households a way to save from one period to the next. We denote assets as a, and next period assets we call $aprime$. Households decisions depend on the (endogenous) state variable a, and one of those decisions is $aprime$. Because decisions now depend on assets, as

well as age, we have $V(a, j)$. The life-cycle value function of the household is now,

$$\begin{aligned} V(a,j) &= \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} + \beta V(aprime, j+1) \\ & \text{if } j < Jr: \; c + aprime = (1+r)a + wh \\ & \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \end{aligned}$$

Notice that current assets a are added to the 'resources' in the budget constraint, as well as interest rate r which is paid on the asset holdings. *aprime*, next period assets subtract from present period assets, and so *aprime* appears on the left-hand-side of the budget constraint: every dollar saved (*aprime*) is a dollar not spent (c).

In terms of the codes this will mean creating a grid for the assets, and adding them into the return function.

2.4 Life-Cycle Model 4: Life-Cycle Profiles

We make no change to the model. Instead we will look at 'life-cycle profiles', which are graphs of how variables change with age. So our value function is still the same,

$$\begin{aligned} V(a,j) &= \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} + \beta V(aprime, j+1) \\ & \text{if } j < Jr: \; c + aprime = (1+r)a + wh \\ & \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \end{aligned}$$

What we want to do is to start households off at age j =1 with zero assets, and watch what they do over their lifetime. Notice that what they do at age 1 is determined by the policy function at age 1, and part of what it tells use is aprime, which in age 2 will become current assets a. So wherever the household starts at age 1, the policy at age 1 tells us where the household is at 2, and then the policy at age 2 tells us where the household is at age 3, etc. We can then draw a graph of what happens to the household over the life-cycle (given how they start) and this will be what we call a 'life-cycle profile'. More precisely, the life-cycle profile of a given variable, say hours worked, shows the mean value of that variable conditional on age (so hours worked at age 1, hours worked at age 2, etc.).[8]

To be able to create life-cycle profiles we need to make an assumption about how agents are 'born' at age $j = 1$. We will assume that they start with zero assets. The codes refer to this as the *jequaloneDist*, the distribution of

[8]Later, in more sophisticated models this distinction will matter and the correct interpretation of the life-cycle profile for a variable is as the age-conditional mean value of that variable (there are also life-cycle profiles for age-conditional standard deviations, etc.). Because the present model has no uncertainty (no idiosyncratic shocks) and all agents are identical at birth (at age $j = 1$) it just so happens that in this model all households will exactly follow this life-cycle profile; there is no variance so everyone is exactly at the age-conditional mean.

agents at age 1.

Because of how the codes work, before we can calculate the life-cycle profiles we have to create something called the 'stationary distribution'.[9] We will not explain the concept of the stationary distribution here, but will cover it later on. To compute the stationary distribution we need one other piece of information, namely how many households are of each age, we will just assume there is an equal fraction of each age, so $1/J$ agents of each age; it is almost always assumed that the total population is normalized to have a mass of 1.[10]

We want to draw three life-cycle profiles: fraction of time worked (h), earnings (wh), and assets (a). To do this we first set up three 'FnsToEvaluate'. And then just run the life-cycle profile command. We then plot these. Note that we are creating 'Mean' life cycle profiles.[11]

2.5 Assignment 1: Add a tax on labor income

Now is a good chance to see if you can make some changes yourself. Try and modify the fourth life-cycle model by adding a tax on labor income, τ_l. So the value function you should implement is,

$$\begin{aligned} V(a,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + \beta V(aprime, j+1) \\ & \text{if } j < Jr: \; c + aprime = (1+r)a + (1-\tau_l)wh \\ & \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \end{aligned}$$

where the only change is the inclusion of $(1-\tau_l)$ multiplying the wh, note that $(1-\tau_l wh)$ is after-tax labor income.

You will need to add the parameter τ_l, which you can give the value $\tau_l = 0.2$ (a 20% tax rate). You will need to modify the value function. You can also try to draw the life-cycle profile of tax paid ($\tau_l wh$).

A code implementing this can be found in the assignments folder: Assignment1_LifeCycleModel.m and the related return fn code.

2.6 Life-Cycle model 5: Earnings are hump-shaped

If we plotted the life-cycle of earnings using real-world data we would see a clear 'hump-shape'. Earning increase when young, peak around age 45-55, and then fall slightly until retirement. We want to capture this fact with our model. To do this we will introduce deterministic labor productivity units as a function of age. The basic idea is

[9] The stationary distribution is not needed for the specific life-cycle profiles we need in this model, but is necessary for other life-cycle profiles in more sophisticated models.

[10] How households are distributed is needed to compute the stationary distribution, but is not relevant to the life-cycle profiles themselves (the life-cycle profile command essentially ignores them). This looks a bit silly in this simple example as we have to set up information we don't need, but it will make sense in more advanced models.

[11] As mentioned life-cycle profiles are more accurately called 'value conditional on age'. We can think of conditional mean, or conditional variance, etc. In the present model because all agents of a given age are identical (because the model is deterministic and all agents start with the same assets, namely zero) only the conditional mean is relevant.

that when young people work one hour, they are less productive/efficient, and so get paid less. This 'hourly labor efficiency' will increase until around 45-55 years old and then decrease.[12]

We will introduce a parameter, κ_j ('kappa'), that is the labor productivity units. Note that this parameter depends on age, so it is a vector of length J, rather than just a single number like all of our previous parameters. The codes will recognise that the parameter is of size J-by-1 and automatically understand that this parameter depends on age and use it appropriately.[13]

The households value function problem becomes,

$$\begin{aligned} V(a,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + \beta V(aprime, j+1) \\ & \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j h \\ & \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \end{aligned}$$

notice that labor earnings is now $w\kappa_j h$, 'wage time hourly productivity times hours'.

To implement this we need to create the parameter κ_j, modify the return function, and also change the FnsToEvaluate for earnings.

If you compare the hours worked profile from this model with that of life-cycle model 4 you can see how this changing hourly wage ($w\kappa_j$) has impacted households decisions to work more/less at different ages.

The purpose of this particular model is really about showing how you can set parameters to depend on age. You can make any parameter depend on age by making it of size J-by-1, and the codes will automatically recognise that the parameter is of this size and treat it as a parameter that depends on age.

2.7 Life-Cycle model 6: Chance of dying

Currently everyone in the model lives to age 100 (period j=81). In reality of course some people die at age 70, and others at age 93, and we would like to add this possibility of dying to our model.[14]

From the perspective of the household we do this by introducing a 'conditional survival probability', s_j, that is the probability of surviving to age $j+1$ given you are age j (or equivalently, it is $1-d_j$, where d_j is the conditional probability of death, the probability of dying this year given you are of age j. We can get these from real world data (many countries calculate them), and we use those for the US from 2010.[15]

[12] The equivalent in the data to find realistic numbers would be to look at how hourly wages change with age.

[13] The codes do not care if it is J-by-1 or 1-by-J, both work fine.

[14] Partly because uncertainty about how long they will live means people save more assets for old age in a manner that is more realistic (relative to just dying earlier, not relative to dying later which is how it will look in our codes relative to the previous life-cycle model), partly because if we want to later turn this model into a model of the economy as a whole, rather than just of a household, then we want a more realistic age distribution (you have probable seen 'demographic pyramid's).

[15] We are calling our agents 'households', but the data is for individuals. We will ignore this distinction here and deal with it in later models where we can think about individuals as distinct from households within the model.

To put the conditional survival probability into the model we assume that households only care about next period if they are alive next period. So they only care about next period with probability s_j. We do this by including s_j as an additional discount factor (that depends on age).

The households problem thus becomes,

$$\begin{aligned} V(a,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + s_j \beta V(aprime, j+1) \\ & \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j h \\ & \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \end{aligned}$$

notice that s_j appears alongside β as an additional discount factor.

We need to add the parameter s_j to the codes (it will be J-by-1 as it depend on age). We also need to tell the codes that s_j should be used as an additional discount factor. Note that there is no change to the return function.

If you compare the asset profiles you will see that people have changed the amount of assets they keep for retirement because there is a chance they will die before getting to consume them.

2.8 Life-Cycle model 7: Warm-glow of bequests

Households in our model currently aim to die with zero assets; this was possible to acheive exactly until we introduced the conditional survival probabilities, you can see it in the life-cycle profiles of assets. In reality many people leave assets behind when they die, and not just people leaving bequests/inheritances to their children, even people without children leave assets, often to charities. One way to deal with this would be to model children explicitly, such 'dynastic OLG' models exists, but are more complex. We will instead model a 'warm glow of bequests': households get utility from leaving assets behind when they die. As a result households with aim to die with non-zero assets, which is realistic.

The warm glow of bequests from leaving behind assets a is, $warmglow(a) = warmglow1 \frac{(a - warmglow2)^{1-warmglow3}}{1-warmglow3}$; where $warmglow2$ is the 'bliss point', which can be thought of as the ideal/target amount of assets to leave behind when dying, and $warmglow1$ determines the importance of the warm glow of bequests (relative to, e.g., consumption or leisure). Notice how the functional form is almost exactly the same as the utility of consumption, and $warmglow3$ acts like the curvature parameter. By using the same functional form it is much easier to choose an appropriate calibration of the parameters for the warm-glow of bequests.

We want agents to only receive the warm glow of bequests when they die. The easiest way to do this would be to only given the warm glow of bequests with assets left at the end of the last period ($j = J$). In this case we would just add a term $\mathbb{I}_{j=J} warmglow(aprime)$ to the return function.[16] Alternatively we could give the warm

[16] $\mathbb{I}_{j=J}$ is the 'indicator function' for $j = J$: it takes a value of 1 when $j = J$ and a value of zero otherwise.

glow of bequests whenever agents die —note that the conditional survival probabilities mean they could die at 'any' time— in which case we could add a term $(1-s_j)warmglow(aprime)$ (to all ages), with $(1-sj)$ being the probability of dying and therefore receiving the warm glow of bequest from the assets left at the end of the period, $aprime$. We will in fact use $\beta(1-s_j)warmglow(aprime)$ as households don't die until the end of the period, but notice that this is really just changing the precise interpretation/value of $warmglow$, nothing meaningful changes when introducing β here. Because agents of all ages have a risk of dying the warm glow would impact asset choices rather directly at all ages, so we will just restrict it to when agents are 75 years or older (retirement age plus 10), which we do by using $\beta(1-s_j)\mathbb{I}_{(j>=Jr+10)}warmglow(aprime)$

So our households problem is now

$$\begin{aligned} V(a,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta\mathbb{I}_{(j>=Jr+10)}warmglow(aprime) \\ & \qquad +s_j\beta V(aprime, j+1) \\ & \text{if } j<Jr: \; c+aprime = (1+r)a + w\kappa_j h \\ & \text{if } j>=Jr: \; c+aprime = (1+r)a + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \end{aligned}$$

notice that the $(1-s_j)\beta warmglow(aprime)$ term is going into the return function.

Implementing this requires us to create the warm glow parameters and then pass these (and β and s_j) into the return function which needs to be modified to include the warm glow of bequests. The warm glow is only given when age j is at years into retirement, this is just done for convenience as otherwise it distorts assets decisions at young ages too much because the risk of dying is non-zero at yound ages; it is close to zero, but not zero.

We will set $warmglow2 = 3$, so the target assets when dying are 3. You can see the impact this has on assets when old by looking at the life-cycle profile of assets.

2.9 Life-Cycle model 8: Idiosyncratic shocks and heterogeneity

Until now the only way households could differ is if they started with different assets at 'birth' ($j=1$). Everything about the model is deterministic, and people in the same state always make the same decisions. We will add 'idiosyncratic shocks', which essentially means that each household will be hit with a shock that affects just them (rather than the economy as a whole; since we do not yet model the economy as a whole the distinction is not yet important, but will be later). We will start with the simplest thing we can, an unemployment shock. The unemployment shock will take two possible values, employed and unemployed. If a household currently has the employed value then they can choose to work just as before. If a household currently has the unemployed value then they are unable to earning income by working (their labor supply, h, must equal zero).[17]

[17] Note that for the retired households these unemployment shocks are irrelevant; we will still model them as that is easier to set up, but we will see in a later model how to change the shocks depending on age (in this case get rid of them when

We will assume that the value of the unemployment shocks next period depend on the value of the unemployment shock today; that the unemployment shocks are a (first-order) markov process.[18] Let z be the unemployment shock. Let $z = 1$ be employment state and $z = 0$ be the unemployment state. So we could think of the possible values of z as $[1, 0]$. We can define the 'markov transition function' as the matrix

$$\pi_z = \begin{bmatrix} p_{ee} & p_{eu} \\ p_{ue} & p_{uu} \end{bmatrix}$$

where p_{ee} is the probability of employment state next period given employment state this period; that is, the probability of *transitioning* from employment this period to employment next period. p_{eu} is the probability of transitioning from employment this period to unemployment next period. Notice then that the first row is the probabilities of the possible outcomes next period, and so the first row must add to one (because tomorrow happens with probability one).[19] The bottom row is the transition probabilities from unemployment this period to employment next period and unemployment next period, respectively.

Because the unemployment shocks are markov the expectations about tomorrow depend on the state today, and as a result the value function will depend on the value of the unemployment shocks.

The household value fn problem is,

$$\begin{aligned} V(a,z,j) = & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\ & + s_j \beta E[V(aprime, zprime, j+1)|z] \\ & \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j zh \\ & \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \\ & zprime = \pi_z(z), \text{ is a two-state markov} \end{aligned}$$

Notice that z is now a state, and also notice that next period value function is now calculated in expectation, the $E[]$ in $E[V(aprime, zprime, j+1)|z]$, because the future is uncertain (it depends on the idiosyncratic shock).

In the codes we will need to let the code know that there is an exogenous state, z, which takes two values, and give the grid and the transition probabilities for z. We also need to modify the return function. Almost everything else is unchanged, the codes recognise the exogenous shock and treat it appropriately when solving the value function, stationary distribution, and life-cycle profiles. We need to make a minor change to *jequaloneDist* as we need to specify whether 'newborns' start in the employed or unemployed state; we will make it 0.7 employed

age is of retirement age).

[18] A (first-order) Markov process is one in which $Pr(y_{t+1}|y_t, y_{t-1}, y_{t-2}, ...) = Pr(y_{t+1}|y_t)$, that is, to predict the value of the markov process next period, y_{t+1}, the only useful information is y_t (if we didn't know the value of y_t other things would be useful, but as long as we know y_t that is all that matters).

[19] This is true of any markov transition matrix: for each row, the sum of the elements in that row much sum to one.

and 0.3 unemployed (remember the total mass of newborns should be 1).

Notice that while both a and z are states, the household can choose a, which is therefore an *endogenous* state. While z cannot be chosen by the household, so it is an *exogenous* state. For exogenous states we have to say how they change over time, which is the role of π_z.

Because adding an idiosyncratic shock z changes the 'size' of the value function and policy function the codes repeat our earlier exercises of plotting these.

2.10 Life-Cycle model 9: Idiosyncratic shocks again, AR(1)

We introduced a shock z that takes two possible values: employed or unemployed. Now let's change z to being about labor productivity. We already have κ_j as deterministic labor productivity depending on age. We will make z an AR(1) process on labor productivity units.[20] The code allows for exogenous shock variables to be any discrete markov process, so we have to convert the AR(1) process z into a markov process. There are many ways to do this, we will use Farmer-Toda; if you are interested in the alternatives like Rouwenhorst, Tauchen and Tauchen-Hussey, see Life-Cycle model A1.

Our household value function is essentially unchanged,

$$\begin{aligned} V(a,z,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta\mathbb{I}_{(j>=Jr+10)}warmglow(aprime) \\ & \qquad +s_j\beta E[V(aprime,zprime,j+1)|z] \\ & \text{if } j<Jr: \; c+aprime=(1+r)a+w\kappa_j zh \\ & \text{if } j>=Jr: \; c+aprime=(1+r)a+pension \\ & 0\leq h\leq 1, aprime\geq 0 \\ & log(zprime)=\rho_z log(z)+\epsilon, \; \epsilon\sim N(0,\sigma^2_{\epsilon,z}) \end{aligned}$$

with the only difference being the definition of z (and implicitly π_z). We could rewrite $log(zprime)=\rho_z log(z)+\epsilon$ as $log(zprime)=\pi_{log(z)}(log(z))$, which is how we can see that an AR(1) is just a specific kind of markov process.

We need a way to discretize $log(z)$, approximating it as a discrete markov process with a grid and transition matrix. When we come to code this we will use the Farmer-Toda method to discretize $log(z)$ and $\pi_{log(z)}(log(z))$. We can then set $z=exp(log(z))$ to get a grid on z, and notice that because we now (after Farmer-Toda method) have $log(z)$ as discrete if follows that $\pi_z=\pi_{log(z)}$; that is we need to take the exponential of the grid, but can just use the transition matrix.[21]

So in the code we just need to create the parameters ρ_z and $\sigma_{\epsilon,z}$. Then use the Farmer-Toda method to create

[20] AR(1) means 'autoregressive process of order 1'. An AR(1) process y_t follows $y_t=(1-\rho)c+\rho+\epsilon_t$, $\epsilon_t\sim N(0,\sigma^2_\epsilon)$. It is a standard model in time-series econometrics. Note that by making the constant $(1-\rho)c$ it follows that $E[y_t]=c$, which gives a nice easy way to interpret c and ρ.

[21] Why the log? Because the AR(1) process can be negative, but by taking the exponential of the AR(1) we guarantee that z is positive (which helps interpret it, what would a negative productivity even mean?).

the grid and transition matrix based on this. Then take exponential of the grid (we will then also normalize the grid to mean of 1, this makes choosing parameter values easier, although it will be irrelevant in our case).[22] We need to decide how many states to use when discretizing the AR(1) process: the more states the more accurate the discretization, but more states will make the code slower. There is no 'correct' number of states, and you should probably check your results for their sensitivity to how you discretize shocks. We will use $n\_z = 21$, which is substantially more than the standard in the literature.

The Farmer-Toda method that we use here is recommended for discretizing AR(1) processes, which the sole exception of those AR(1) processes with very high persistence, $\rho_z \geq 0.99$, for which the Rouwenhorst method is recommended. The Rouwenhorst method, along with other methods common in the literature like the Tauchen method and the Tauchen-Hussey method can be easily implemented, see Life-Cycle Model A1, but I recommend against using them as they are not as accurate.

A very brief explanation of how Farmer-Toda works: in a first step, a grid for z is set using either evenly-spaced points. The second step is to choose the transition probabilities. For a given grid point today the corresponding row of the transition matrix is a probability distribution for tomorrows state, so Farmer-Toda choose the transition probabilities of this row to target the moments of this probability distribution (by default the first two moments, but code allows up to four). If we have, e.g., five grid points, then this is two restrictions, which would not be enough (there would be a continuum of possible solutions), so Farmer-Toda add the target of maximizing the entropy (relative likelihood) which makes the solution unique. Note that this is done row-by-row for the transition matrix.[23]

The main weakness of Farmer-Toda is that it takes about 1 second to run, while the alternatives take more like 1/1000th of a second. For our purposes this is irrelevant but may matter if you want to do, e.g., simulated likelihood estimation or simulated method of moments estimation of the life-cycle model. (Don't worry if you don't understand what these are, they are much more advanced than our current focus.)

[22] We discretize the AR(1) and then take the exponential of the grid, and then normalize the grid to have a mean of one. Notice that we could alternatively just discretize the AR(1) and then use $exp(z)$ in the return function (in the budget constraint). This would run fine, but it would not have the mean of $exp(z)$ being one. This would be massively complicate things like trying to calibrate/estimate κ_j to the data.

[23] The idea of matching the moments and then using maximum entropy is actually from Tanaka and Toda (2013) who apply it to i.i.d random variables, Farmer and Toda (2017) is about generalizing to a markov process by doing this row-by-row for the transition matrix.

2.11 Life-Cycle model 10: Exogenous labor supply

We solve a version of Life-Cycle model 9 in which the labor supply is exogenous, that is there is no choice of hours worked,

$$\begin{aligned} V(a,z,j) &= \max_{c,aprime} \frac{c^{1-\sigma}}{1-\sigma} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\ &\qquad + s_j \beta E[V(aprime, zprime, j+1)|z] \\ &\text{if } j < Jr: \ c + aprime = (1+r)a + w\kappa_j z \\ &\text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\ &aprime \geq 0 \\ &log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

notice that h has disappeared from the 'decision variables', and so has the disutility of hours worked. z is now a 'stochastic endowment' or 'exogenous labor income' or 'exogenous earnings' (more precisely $w\kappa_j z$ is the endowment/labor income endowment); just different ways to describe it. This problem is often called the '*income flucations*' problem in the literature.

Implementing this first sets $n_d = 0$ and is then essentially just setting the 'decision variable grid' to be empty; $d\_grid = []$, and deleting h from all the formulaes (as well as removing the parameters that related to hours worked in either the utility function or the budget constraint). You could do the same by setting $d\_grid = 0$.

Note that keeping w is slightly odd here, but it makes it easier to compare results to the endogenous labor supply models. In any case $w = 1$ so that is irrelevant.

2.12 Life-Cycle model 11: Idiosyncratic shocks again, persistent and transitory

We introduced an idiosyncratic shock to labor productivity units. We modelled that shock as an AR(1) process, which was discretized and entered the model as a markov exogenous state. But empirical work suggests we can do better by modelling changes in labor producitivity units as a combination of two shocks, one persistent and one transitory. The persistent shocks we will model as an AR(1). The transitory shock we will model as i.i.d.

We can include an i.i.d exogenous state, which VFI Toolkit calls 'e', into our model. This will be the first thing we have seen that takes us beyond the 'baseline setup' of VFI Toolkit which has a model 'action space' being $(d, aprime, a, z, ..)$, we have seen in our models up until now. Because we go beyond the baseline we need to use *vfoptions* and *simoptions* to tell VFI Toolkit that we are using an i.i.d. exogenous state, e.

The persistent shocks, z, we will model as an AR(1). The transitory shock, e we will model as i.i.d. Because VFI Toolkit will know that z is markov and e is i.i.d. it can take advantage of the simplicity of the i.i.d. shock to run faster. Because this increases the 'action space' of our model, we will have to modify *ReturnFn* and *FnsToEvaluate* appropriately.

Our household value function now has ze in place of z to determine idiosyncratic producitivity units,

$$\begin{aligned} V(a,z,e,j) &= \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta\mathbb{I}_{(j>=Jr+10)}warmglow(aprime) \\ &\quad +s_j\beta E[V(aprime,zprime,eprime,j+1)|z] \end{aligned}$$

$$\text{if } j<Jr: \ c+aprime=(1+r)a+w\kappa_j zeh$$

$$\text{if } j>=Jr: \ c+aprime=(1+r)a+pension$$

$$0\leq h\leq 1, aprime\geq 0$$

$$log(zprime)=\rho_z log(z)+\epsilon, \ \epsilon\sim N(0,\sigma^2_{\epsilon,z})$$

$$log(e)\sim N(0,\sigma^2_e)$$

Notice that z is AR(1) and e is i.i.d. normal (in logs).

Note that to get an exogenous shock (log) e which is i.i.d. $N(0,\sigma^2_e)$, we can simply use the same method as we did for the AR(1) process and just set $\rho=0$, because it is explicitly i.i.d. we then keep only the first row (or any row) of the transition matrix created to store as a column vector in pi_e.

How do we let the codes know we have two exogenous shocks? First we just set up z exactly as we would if it were the only such shock (because it is the only markov shock). Then we set up e in a similar fashion, except that it has to go in $vfoptions$ and $simoptions$ (because it is not part of the baseline setup). So we set $vfoptions.n\_e$, $vfoptions.e\_grid$ and $vfoptions.pi\_e$ to be the number of points, the grid, and the probability weights, respectively. We need to put the exact same things into $simoptions.n\_e$, $simoptions.e\_grid$ and $simoptions.pi\_e$. The important difference when setting up an i.i.d. 'e' variable compared to a markov 'z' variable is that instead of having a transition matrix 'pi_z' we instead have probability weights 'pi_e'; instead of a matrix we have a column vector.

2.13 Alternative Exogenous Shock Processes (Appendix A)

There are many other things we might want to do with exogenous shocks, both in terms of different kinds of shock process —AR(1) with parameters that depend on age, AR(1) with gaussian-mixture shocks, VAR(1)—- as well as alternative methods for discretizing shocks —Tauchen, Rouwenhorst, Tauchen-Hussey. A wide variety of these are demonstrated in Life-Cycle Models Appendix A. It also explains all the types of shocks VFI Toolkit recognizes, namely markov exogenous state z, and i.i.d. exogenous state e, which we have seen, plus semi-exogenous states which won't be seen until Life-Cycle model 29. Additionally it explains how to set up multiple shocks, so for example you might have two markov and three i.i.d. shocks. How to handle that the shocks are correlated, or that the transitions of one shock depend on the other, are also all explain in the various Appendix models. So go check out Life-Cycle Models Appendix A for information on how to discretize and handle a wide variety of shock types.

2.14 Assignment 2: Alternative Utility Functions

Because the utility function is coded inside the return function changing it is easy. We will now present a few different utility functions. It is left as an assignment for you to try and implement the 'non-seperable CES utility function' by modifying LifeCylceModel9.m code, and look at how they change the life-cycle profiles. A 'solution' is provided as Assignment2_LifeCycleModel.m.

We can write a generic utility function of consumption and leisure as $u(c,l)$, where leisure can be defined as $l = 1 - h$, is the fraction of time not worked. The advantage of using leisure rather than hours worked in the utility function is that utility is increasing in leisure, which makes theoretical properties easier to derive; we will no longer need to subtract the disutiliy of hours worked as we did until now.

Using this generic utility function we can write the value function problem as,

$$\begin{aligned} V(a,z,j) \quad = \quad & \max_{c,aprime,l} u(c,l) + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\ & \qquad +s_j \beta E[V(aprime, zprime, j+1)|z] \\ & \text{if } j < Jr: \ c + aprime = (1+r)a + w\kappa_j z(1-l) \\ & \text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\ & 0 \leq l \leq 1, aprime \geq 0 \\ & log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

notice that $h = 1 - l$ has been substituted into the budget constraint, and the restriction that $0 \leq h \leq 1$ has been replaced with $0 \leq l \leq 1$; the obvious modifications around replacing $h = 1 - l$ in the FnsToEvaluate also need to be made.

In the Life-Cycle models we have seen until now we used the 'seperable CES' utility function,

$$u(c,l) = \frac{c^{1-\sigma_c}}{1-\sigma_c} + \psi \frac{l^{1-\sigma_l}}{1-\sigma_l}$$

note of course that this is not the exact form of the utility function we used until now, but it has the same properitie, just that we are now writing it in terms of l instead of h (you could substitute $1 - h$ in place of l here and the properties of the utility function would be the same).[24]

The CES refers to 'constant elasticity of substitution', and the 'seperable' refers to how the (marginal) utility of consumption is independent of leisure (and vice-versa the marginal utility of leisure is independent of consumption).

[24] The exact numbers of utils would differ, but as is well know utility is only defined up to a linear transformation; that is, the utility functions $u()$ and $a + bu()$, where a,b are constants, both represent the same preferences. This can be proven in very general terms, see any advanced Microeconomic Theory textbook. If you are unconvinced, write out a basic problem, say one-period lagrangian problem choosing between consumption of two goods, and solve it with both these utility fns, you will see how a disappears when you take derivatives, and then b cancels out because it is in all the marginal utility terms.

One important alternative is 'non-seperable CES' utility, which is given by,

$$u(c,l) = \frac{(\sigma_1 c^{\sigma_2} + (1-\sigma_1) l^{1-\sigma_2})^{1-\sigma_3}}{1-\sigma_3}$$

which also has constant-elasticity-of-substitution, but now the marginal utility of consumption depends on leisure, and vice-versa (take the derivatives and you will see they depend on both). σ_1 can be understood as a 'share' parameter, in this case it is the share of consumption (versus leisure) in utility. $1/(1-\sigma_2)$ is the elasticity of substitution between c and l. σ_3 is determining the decreasing marginal utility, and will therefore determine things like risk-aversion. Note that you can obviously set $h = 1 - l$ and solve the model in terms of hours worked, the 'assignment solution' does this: modify life-cycle model 9 to have non-seperable CES utility function.

An important subcase of the 'non-seperable CES' utility function, is given in the limit when $\sigma_2 \to 0$, and we get,

$$u(c,l) = \frac{(c_1^{\sigma} l^{1-\sigma_1})^{1-\sigma_3}}{1-\sigma_3}$$

This is particularly important in models that contain economic growth; see Life-Cycle model 22 for an explanation of why it matters, and of how to include deterministic growth in a model.[25]

An alternative non-seperable utility function is GHH preferences.[26] They are easier to write in terms of hours worked,

$$u(c,l) = \frac{\left(c - \psi \frac{h^{1+\eta}}{1+\eta}\right)^{1-\sigma}}{1-\sigma}$$

what matters to labor supply is just the wage. As the marginal rate of substitution is independent of consumption and only depends on the real wage, there is no wealth effect on the labour supply. They are not consistent with a balanced growth path (that would result from deterministic growth). A version that extends this to allow a balanced growth path are Jaimovich-Rebelo preferences.

2.15 Life-Cycle model 12: Epstein-Zin preferences

Epstein-Zin preferences avoids an 'restriction' implicit in the use of CES utility functions (with von-Neumann Morgenstern expected utility), namely that a single parameter determines both risk aversion and the intertemporal elasticity of substitution. Epstein-Zin preferences have an additional parameter than modifies the degree of risk aversion.

Implementing Epstein-Zin preferences involves a reformulation of the value function problem itself. In this model we will use 'Epstein-Zin in utility units', note that traditionally Epstein-Zin was done in 'consumption units'; Appendix B.1 explains these, and how you can use the consumption-units version by setting *vfoptions*. The

[25] You could call this the Cobb-Douglas utility function. It is the same functional form as the Cobb-Douglas production function, just that the parameters need some rewriting to make it obvious.

[26] Greenwood-Hercowitz-Huffman preferences.

other issue with Epstein-Zin preferences is that we have to be careful how we handle warm-glow of bequests.

Starting from Life-Cycle model 9, we make just the change to Epstein-Zin preferences and get the value function problem,

$$\begin{aligned} V(a,z,j) &= \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} \\ &\quad -\beta E[-s_j V(aprime, zprime, j+1)^{1-\phi} - (1-s_j)G(aprime)^{1+\phi}|z]^{\frac{1}{1+\phi}} \\ &\text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j zh \\ &\text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\ &0 \leq h \leq 1, aprime \geq 0 \\ &log(zprime) = \rho_z log(z) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

where $1-\phi$ is modifying the amount or risk-aversion. The minuses before and inside the expectation are needed to handle that the utility function is negative. A larger value of ϕ is associated with a higher risk aversion ($\phi > 0$ corresponds to more risk aversion that standard vonNeumann-Morgenstern preferences, when $\phi = 0$ we get standard vonNeumann-Morgenstern preferences). This setup depends on the fact that we have $u < 0$, if $u > 0$ the specification is different, see Appendix B.1.

Implementing this in the codes is as simple as naming the additional parameters, and setting,
$vfoptions.exoticpreferences =' EpsteinZin'$
You specify the names of the parameters that modifies the Epstein-Zin preferences using *vfoptions.EZriskaversion='phi'*. Because the specification depends on the sign of the utility function, you also need to set *vfoptions.EZpositiveutility*.

Note that with Epstein-Zin preferences we want to seperate out the conditional survival probabilities from the discount factor (this is implicit in the above formulation as s_j does not appear everywhere β appears). You can do this when using Epstein-Zin preferences by setting *vfoptions.survivalprobability='sj'* (and not including it in the *DiscountFactorParamNames*).

When using Epstein-Zin preferences we have to be careful about how we treat warm-glow of bequests. VFI Toolkit handles this for you, and you just set up *vfoptions.WarmGlowBequestsFn* which depends on next-period endogenous state (on *aprime*). The idea is to use the same thing as your utility function. In this model the utility of consumption is $\frac{c^{1-\sigma}}{1-\sigma}$, and so we use the warm-glow of bequests function $\frac{aprime^{1-\sigma}}{1-\sigma}$. We multiply this by a constant wg to determine the importance of bequests, and we also use a term so that bequests are only received ten periods after retirement (in line with previous models). Hence our final warm-glow of bequest function is $\mathbb{I}_{(agej \geq Jr+10)} wg \frac{aprime^{1-\sigma}}{1-\sigma}$. Here wg controls the strength of the bequest motive. For an explantion see Appendix B.1.5 which explains how to set up the bequests for Epstein-Zin preferences, as well exactly what the codes are doing.

2.16 Life-Cycle model 13: Simulating Panel Data

We used the FnsToEvaluate to create the life-cycle profiles. We can also use the same FnsToEvaluate to simulate panel data (now that we have idiosyncratic shocks). Note that a single household can be simulated to create a time series, and by starting numerous household simulations we get panel data.

We will use the model of Life-Cycle model 9, so we do not repeat that here. And we will simulate the same FnsToEvaluate as we used in that model.

We can choose the number of time periods for the panel data simulations using *simoptions.simperiods* (it cannot be larger than number of model periods in a life-cycle model), and can choose the number of individuals time series simulations using *simoptions.numbersims* (1000 by default). Any simulation needs to start from somewhere. The codes use 'InitialDist' as a distribution from which the starting state of agents is drawn (*seedpoint* is what VFI Toolkit calls it internally), we will just set InitialDist to be equal to the stationary distribution in the current example, but in principle any distribution could be used.

2.17 Life-Cycle model 14: More Life-Cycle Profiles

Now that our models have idiosyncratic shocks so we can look at life-cycle profiles other than the mean. For example we will look at the variance conditional on age. We will also see how to group ages together in 'age bins', for example we can look at mean income for 5-year age-groupings, or at the labor earnings Gini coefficient for all 'working age' individuals. This will all take just a few lines of code.

The model is exactly that of Life-Cycle model 9, so we do not repeat that here.

The variance, gini, and much more conditional on age were already being automatically computed, just that we did not look at them.

To use 'age bins' we just need to specify them. By default it is assumed that every period is it's own bin, this would correspond to setting $options.agegroupings = 1 : 1 : N_j$. To compute them in 5 year age bins use $options.agegroupings = 1 : 5 : N_j$ (note, if N_j is not divisible by 5 the last bin will contain less than 5). For working age as a single bin use $options.agegroupings = [1, Params.Jr]$. In general, *options.agegroupings* is a vector each element of which defines the period at which a new bin starts. So to give a more complex example $options.agegroupings = [1, 7, 10, 11, 15]$ would create four bins, the first would be periods 1 to 6, the second is periods 7 to 9, third is 10, fourth is 11 to 14, fourth is 15 on (until N_j, inclusive). We then just add *options* as the final input to the Life Cycle Profiles command.

3 Illustrating some important concepts

3.1 Life-Cycle model 15: Consumption and Borrowing Constraints 1

We will use our models to look at how borrowing constraints impact consumption when young. Households would like to smooth consumption over their lifetime (based on life-cycle permanent income hypothesis). But earnings have a hump-shape. So households want to save some income from the 'hump' to consume in retirement, which is easy enough. They also want to bring some income from the 'hump' forward to when they are young, which requires borrowing. But they are unable to borrow due to borrowing constraints when young. So consumption is lower when young, and more importantly the marginal utility of consumption when young is higher, and marginal propensities to consume are also higher when young.

We will use a simple life-cycle model with exogenous labor, no idiosyncratic shocks, and we will make the life-cycle profile of labor producitivity units very step to really make the borrowing constraint bind strongly when young.

$$
\begin{aligned}
V(a,j) \quad = \quad & \max_{c,aprime} \frac{c^{1-\sigma}}{1-\sigma} + \beta V(aprime, j+1) \\
& \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j \\
& \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\
& 0 \leq h \leq 1 \\
& aprime \geq borrowingconstraint
\end{aligned}
$$

notice that labor earnings is now $w\kappa_j$, which is exogenous so you could equally call it 'endowment' or exogenous income.

The code sets a parameter called 'borrowconstraint' that you can change to see how the borrowing constraint matters. There are life-cycle profiles of consumption and marginal utility consumption and you can see how the borrowing constraint causes a 'jump' in consumption because people cannot borrow and consume as much as they would like. You can also see that the borrowing constrained have a very high marginal utility of consumption. As a result there are large welfare gains for the household from anything that loosens the borrowing constraint (whether it is being able to borrow more, lump-sum transfers, etc.). The borrowing constrained household would also consume any extra income and so has a high marginal propensity to consume (how high depends on how long borrowing constraint is expected to continue binding).

Note that all of our models until now have had a borrowing constraint, namely $aprime \geq 0$. This was enforced via the grid on assets which had a minimum value of zero, meaning that choosing less than zero assets was simply not possible. To implement the present model we have to set a lower minimum point on the grid on assets, and we could either use that implicitly, or do as the codes have here and add a parameter called 'borrowconstraint' that

imposes a tighter borrowing contraint via the return function (note that the parameter must be greater than or equal to the minimum point on the asset grid else it is not relevant). It is worth mentioning that it is 'not possible' to have a model without a borrowing constraint, in the sense that in the absence of a borrowing constraint optimial behaviour would be to borrow an infinite amount, consume it all, and die in infinite debt; mathematically this is of course possible, I say not possible in the sense you would never actually want to use that model.

3.2 Life-Cycle model 16: Consumption and Borrowing Constraints 2

We will look at how stochastic shocks mean borrowing contraints can sometimes bind when shocks are 'bad', and what this means for consumption. We use Life-Cycle Model 9, which had an AR(1) idiosyncratic shock process, except we will use just 5 states to discretize it so as to make it easier to see 'bad' shocks.[27]

In stochastic models borrowing constraints often bind after a series of 'bad' shocks. They don't bind most of the time, nor for most households, so looking at life-cycle profiles is not a good way to see them. Two options are possible: (i) we could simulate households that get a series of bad shocks and see them run up against the borrowing constraints, or (ii) we could run regressions. We will take the first approach here.

For the panel data, we first simulate a panel data set and then look for candidate households for whom the borrowing constraints are likely to bind, that is, households who have a run of bad shocks. To make it clearer we first look for households that have median or better shocks in the first 15 periods, so that they mostly get away from the borrowing constraints, and then narrow our search to those who have 7 or more bad shocks in periods 16-25. This defines our candidate households. We plot the first 30 periods for ten of these candidate households (the ten are arbitrarily selected from among all candidate households) and for these we plot the consumption, the marginal utility of consumption, and next period assets. You can see in Figure 3 how the households during periods 16-25 run into the borrowing constraint as a result of the bad shocks, and as a result their consumption is low and their marginal utility of consumption is high.

Note that the reason households do not like running into borrowing constraints —because it forces them to cut consumption more than they would otherwise like— is exactly the same in this model as in the previous Life-Cycle Model 15, what differs is just the reason that the borrowing constraints bind.

To emphasise that the borrowing constraints bind for a different reason to what happened in Life-Cycle Model 15 we flatten the earnings profile to largely eliminate what drove them in that model.

3.3 Life-Cycle model 17: Precautionary Savings (with exogenous earnings)

We have already seen how the binding of borrowing constraints can force households to reduce consumption, and in particular how this leads to a large increase in the marginal utility of consumption. Households want to avoid

[27] Just 5 points will provide a poor approximation to the AR(1) process, we use just five points purely to make it easier to illustrate.

this situation of binding borrowing constraints, and so they engage in 'precautionary savings': savings that are 'higher' than they would otherwise be so as to avoid the probability of the borrowing constraints binding.[28]

We use the exact same model from Life-Cycle Model 10, and then solve it a second time, which we call the 'no shock' version, in which $z = 1$ (rather than an AR(1) process in logs). We will use just 3 states to discretize it so as to make it easier to see how the savings policies differ.[29]

How to see precautionary savings? We take three approaches: (i) savings policy functions, (ii) life-cycle profiles, (iii) aggregate assets.

We start with the savings policies. Figure 1 shows how households in the model with the median shock value and low asset holdings[30] choose to save more, for precautionary reasons, than households in the model without shocks. This is true even though the median shock is actually slightly lower earnings than the model without shocks (0.9976 vs 1). The top panel of Figure 2 shows the period before retirement in which there are no future shocks and so no precautionary savings motive,[31] you can see that now the households with the median shock actually save less than those with no shocks (because of the 0.9976 vs 1 earnings mentioned before). The other panels of Figure 2 show that in retirement, because shocks are now irrelvant the households all make the exact same decisions (multiple lines are plotted, but you can only see one as they are all on top of each other).

We look at life-cycle profiles in Figure 3. Those on the left are with shocks, on the right are without shocks; except the bottom row that shows those with shocks minus those without shocks. Looking at the bottom-left panel we confirm that there is no difference in mean earnings for households in the models with and without shocks (is just the left panel of top row minus the right panel of top row); this is just to confirm that any savings differerences are not coming from differences in mean earnings. In the bottom-right panel we see the precautionary savings: households with shocks have higher assets than households without shocks. We can also see how the precautionary savings interact with the life-cycle consumption-smoothing motives we discussed in Life-Cycle Model 15. When households are young the life-cycle consumption-smoothing motives dominate and no households save any assets (see panels in second row); the precautionary savings motives are there, but they are overwhelmed. Later in working age the life-cycle consumption-smoothing motives largely disappear (in sense of keeping assets at zero) and we can see the precautionary savings in the bottom-right panel with households that face earnings risk holding more assets tha households that do not face any shocks.

Our final look at precautionary savings comes from looking at aggregate assets —the total assets held by all

[28] Note that precautionary savings is a phenomenon of stochastic models: precautionary savings are savings to avoid the chance of a borrowing constraint binding that the household. In a deterministic model households know for a fact that borrowing constraints will or will not bind. In a stochastic model a household can do precautionary savings, get good shock outcomes that means the household never end up in a situation where the borrowing constraint would bind, and now have ended up saving more than they would would like ex-post. In a deterministic model it is not possible to choose savings optimally ex-ante that you end up regretting as overly cautious ex-post.

[29] Just 3 points will provide a poor approximation to the AR(1) process, we use just three points purely to make it easier to illustrate. We do make one other change from Life-Cycle Model 10, namely we make the initial age j=1 distribution of agents have zero assets (as before) and the stationary distribution of shocks (previously just the median), this is so the life-cycle profile of mean earnings is exactly equal in the models with and without shocks; the difference is very small.

[30] Note that the graph zooms in on low asset holdings, as seen by the x-axis.

[31] Shocks are only relevant during working age as they are to labor producitivity units, and retirees never work.

households added up across the stationary distribution— where we can see that assets are higher for the households with shocks than for the households without shocks. We don't use these 'aggregates' much in life-cycle models as they make more sense when thinking about the economy as a whole, but this idea that households will with precautionary savings save up more assets is very important in general equilibrium models of incomplete markets, an idea we will discuss later. The aggregate assets are printed to the screen when you run the codes.

3.4 Life-Cycle model 18: Precautionary Savings with Endogenous labor

Precautionary savings are one way to avoid the binding of borrowing constraints. An alternative is to adjust labor supply; working more to avoid hitting borrowing constraints. So models with endogenous labor supply have lower amounts of precautionary savings. We will use Life-Cycle Model 9, but discretizing AR(1) process on z using just three shocks values to make things clearer.

Note that while endogenous labor will reduce precautionary savings this is not the only effect it will have on savings, because it will also interact with life-cycle motives for savings; for example, households can choose to work more when they reach higher earnings levels during middle age and do all their saving for retirement then, which would do some combination of reducing assets when young (although most households don't save when young and are already up against their borrowing constraints) and increasing the asset levels in middle-age that they carry over into retirement. As a result whether endogenizing labor supply results in higher or lower levels of assets overall depends on various factors; especially, the utility of leisure and the form of the utility function.

Too see all of this we will solve the model four times. First we solve the model with endogenous labor and exogenous shocks, second we solve the model with endogenous labor and no shocks, third we solve the model with exogenous labor and exogenous shocks, and fourth we solve the model with exogenous labor and no shocks.

Comparing the model with endogenous labor and shocks to the model with endogenous labor without shocks we can see that there are precautionary savings as a result of shocks, seen, e.g., in the higher aggregate capital/income ratio in the model with shocks (see the numbers printed to screen). We can also see how agents near the borrowing constraint (low assets) chose to work more hours as a way to avoid the budget constraint. In period $j = 1$ households are trying to escape the borrowing constraint mostly for life-cycle reasons, so there is little difference between the models with and without shocks in the top panel of Figure 4. By constrast by age $j = 20$ the borrowing constraints are mostly about bad shocks, and so we can see how the model with shocks alway results in more labor supply at low assets levels compared to the model without shocks (life-cycle motives also play a role). You can see how this use of precautionary labor supply is often enough to actually reduce savings relative to the model without shocks; top panels of Figures 1, 2 and 3. This is partly because precautionary labor reduces demand for precautionary savings, and partly because of how it changes life-cycle motives.

If we now compare the model with endogenous labor and exogenous shocks to the model with exogenous labor and exogenous shocks we can see that in our current calibration endogenizing labor leads to an increase in savings; e.g., by comparing capital/income ratios. Note that this varies over the life-cycle, you can see how endogenous

labor supply decreases savings at ages $j = 1$ (bottom panel of Figure 1) and $j = 20$ (bottom panel of Figure 2), but for life-cycle motives actually increases savings at age $j = 40$ (bottom panel of Figure 4). This is because the reduced precautionary savings are more than offset by the increased saving for life-cycle motives at young ages, but then the life-cycle motives go towards higher savings, adding to the precautionary savings motive, at middle age. This provides a good example of how there are often various effects of any model changes working in different directions and it is rarely clear beforehand which will dominate.

To ease the comparison of results between the model with endogenous labor supply and that with exogenous labor supply we have normalized earnings in the later so that both have the same aggregate (mean) earnings. The reason is as follows: note that for exogenous labor earnings is $w\kappa_j z$, while with endogenous labor earnings is $w\kappa_j zh$, and $0 \leq h \leq 1$, so earnings will be lower in the model with exogenous savings.[32] We can do this be simply adding a parameter to earnings in the exogenous labor model, so that earnings are now $\phi_{normalizemeanearnings} w\kappa_j z$. By setting $\phi_{normalizemeanearnings}$ equal to the mean earnings in the model with endogenous labor divided by the mean earnings in the model with exogenous labor we will give both the models the same mean earnings.[33]

This exercise of endogenous labor supply reducing precautionary savings is much cleaner in an infinite-horizon (and general equilibrium) model (Pijoan-Mas, 2006), where there are no life-cycle considerations.

3.5 Life-Cycle model 19: Incomplete Markets

We saw how borrowing constraints and idiosyncratic shocks together lead to precautionary savings. There is an important third aspect to this that was 'hidden' in the background: incomplete markets. Loosely speaking, complete markets is the presence of perfect insurance which all households buy and therefore the idiosyncratic shocks they later receive are irrelavant as they are perfectly insured against them. With complete markets, because households perfectly insure themselves, there is no meaningful inequality; no meaningful distribution of households.

If we wanted to create a life-cycle model in which there was an exogenous shock that takes two values we would need two assets (one of which pays out in the case the shock takes the first value, the second asset pays out in the case the shock takes the second value), and so we won't attempt to do that here.

Instead we will solve a two period model and show how briefly the idea of how complete and incomplete markets relate. We start with a deterministic two-period model, then add shocks in the second period. We end with a brief discussion of how Representative Agents are related to complete markets models. Note that all the models being solved elsewhere in this document (and more generally, in essentially all of the heterogenous agent model literature) are incomplete markets models.

Let's start with a simple two-period deterministic model. The household solve the following maximization

[32] We could try and deal with this by looking at aggregate capital/income ratios, but this is not a full fix as there would still be income effects that may be playing a substantial role. By normalizing mean earnings we substantially reduce these other channels.

[33] Because we are doing this to the model with exogenous labor supply we don't need to worry about how people react in determining mean earnings. To find this $\phi_{normalizemeanearnings}$ we would first just solve both models with $\phi_{normalizemeanearnings} = 1$ and then calculate the appropriate $\phi_{normalizemeanearnings}$.

problem,

$$\begin{aligned} \max_{\{c_1,c_2,a\}} \; & u(c_1) + \beta u(c_2) \\ \text{s.t. } & c_1 + a = y_1 \\ & c_2 = (1+r)a + y_2 \end{aligned}$$

We can solve this using the Lagrangian method. First write the Lagrangian,

$$\mathcal{L}(c_1, c_2, a, \lambda_1, \lambda_2) = u(c_1) + \beta u(c_2) + \lambda_1(y_1 - c_1 - a) + \lambda_2((1+r)a + y_2 - c_2)$$

The first-order conditions of this are

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial c_1} = 0; \quad & u'(c_1) + \lambda_1(-1) = 0 \\ \frac{\partial \mathcal{L}}{\partial c_2} = 0; \quad & \beta u'(c_2) + \lambda_2(-1) = 0 \\ \frac{\partial \mathcal{L}}{\partial a} = 0; \quad & \lambda_1(-1) + \lambda_2(1+r) = 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda_1} = 0; \quad & c_1 + a = y_1 \\ \frac{\partial \mathcal{L}}{\partial \lambda_2} = 0; & c_2 = (1+r)a + y_2 \end{aligned}$$

From the first three of these we get

$$u'(c_1) = \beta(1+r)u'(c_2)$$

Purely to make it easier to see how this compares to what we will do next let $\beta(1+r) = 1$ to get,[34]

$$u'(c_1) = u'(c_2)$$

Because there is one asset it is possible to shift consumption between the two periods and equate the marginal utitlities. Complete markets are key to this as we will see in the next two examples.

Now, we will introduce a shock, z, in the second period that takes on of two possible values z_1 or z_2. Let p be the probability of $z = z_1$, so then $1 - p$ is the probability of $z = z_2$. What we are about to solve is an *in*complete

[34] In a Representative Agent model it is typically true that $\beta(1+r) = 1$ in general equilibrium.

market model; we will explain later why this is incomplete markets. The household problem is,

$$\begin{aligned}\max_{\{c_1,c_2(z_1),c_2(z_2),a\}} u(c_1)+\beta[pu(c_2(z_1))+(1-p)u(c_2(z_2))]\\ \text{s.t. } c_1+a=y_1\\ c_2(z_1)=(1+r)a+y_2(z_1)\\ c_2(z_2)=(1+r)a+y_2(z_2)\end{aligned}$$

where $c_2(z_1)$ is consumption in the second period when $z=z_1$, $c_2(z_2)$ is consumption in the second period when $z=z_2$. Notice that the difference caused by the shock is that the income/endowment is $y_2(z_1)$ vs $y_2(z_2)$; so we can think of the shock as being low/high earnings. We will assume, without loss of generality, that $y_2(z_1)<y_2(z_2)$. Notice how the asset a allows us to shift consumption between period one and period two, but there is no way to shift consumption between the two states/shocks in period two; this is the incomplete markets. Writing the Lagrangian we get,

$$\begin{aligned}\mathcal{L}(c_1,c_2(z_1),c_2(z_2),a,\lambda_1,\lambda_2,\lambda_3)=u(c_1)+\beta[pu(c_2(z_1))+(1-p)u(c_2(z_2))]+\lambda_1(y_1-c_1-a)\\ +\lambda_2((1+r)a+y_2(z_1)-c_2(z_1))+\lambda_3((1+r)a+y_2(z_2)-c_2(z_2))\end{aligned}$$

The first-order conditions of this are

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial c_1}=0;\quad u'(c_1)+\lambda_1(-1)=0\\ \frac{\partial\mathcal{L}}{\partial c_2(z_1)}=0;\quad p\beta u'(c_2(z_1))+\lambda_2(-1)=0\\ \frac{\partial\mathcal{L}}{\partial c_2(z_2)}=0;\quad (1-p)\beta u'(c_2(z_2))+\lambda_3(-1)=0\\ \frac{\partial\mathcal{L}}{\partial a}=0;\quad \lambda_1(-1)+\lambda_2(1+r)+\lambda_3(1+r)=0\\ \frac{\partial\mathcal{L}}{\partial\lambda_1}=0;\quad c_1+a=y_1\\ \frac{\partial\mathcal{L}}{\partial\lambda_2}=0;c_2(z_1)=(1+r)a+y_2(z_1)\\ \frac{\partial\mathcal{L}}{\partial\lambda_3}=0;c_2(z_2)=(1+r)a+y_2(z_2)\end{aligned}$$

From the first, second, third and fourth, we get,

$$u'(c_1)=(1+r)\beta[pu'(c_2(z_1))+(1-p)u'(c_2(z_2))]$$

Again, purely to simplify to something easier to look at, let $\beta(1+r)=1$, to get,

$$u'(c_1)=pu'(c_2(z_1))+(1-p)u'(c_2(z_2))$$

so with incomplete markets we have that the household is able to shift between period one and period two consumption using the asset, a, and so it sets the marginal utility of consumption in period one equal to the expected marginal utility of consumption in period two (expected=sum weighted by probabilities). Note that we had $y_2(z_1) < y_2(z_2)$ and so it follows from the period 2 budget constraints that $c_2(z_1) < c_2(z_2)$ (as a must be the same for both), from which if follows that $u'(c_2(z_1)) > u'(c_2(z_2))$ (because u is increasing and concave; standard assumptions that $u' > 0$ and $u'' < 0$; note that the later says that there is decreasing marginal utility). Combining $u'(c_2(z_1)) > u'(c_2(z_2))$ with $u'(c_1) = pu'(c_2(z_1)) + (1-p)u'(c_2(z_2))$ we get that $u'(c_2(z_2)) < u'(c_1) < u'(c_2(z_1))$ (take the 'weighted sum'/'average' of any two positive numbers and you will get something that is inbetween them).

Now we will look at the two period model with a shock in the second period that can take two values, again. But this time we will solve the complete markets version. For complete markets we will need two assets, with different returns depending on states, and the easiest way to do this will be to have asset a_1 which returns $(1+r)a_1$ when $z = z_1$ and zero when $z = z_2$, as well as asset a_2 which returns zero when $z = z_1$ and $(1+r)a_2$ when $z = z_2$.[35] The household problem is,

$$\begin{aligned}\max_{\{c_1, c_2(z_1), c_2(z_2), a_1, a_2\}} & \; u(c_1) + \beta[pu(c_2(z_1)) + (1-p)u(c_2(z_2))] \\ \text{s.t. } & \; c_1 + a_1 + a_2 = y_1 \\ & \; c_2(z_1) = (1+r)a_1 + y_2(z_1) \\ & \; c_2(z_2) = (1+r)a_2 + y_2(z_2)\end{aligned}$$

Note that, e.g., the zero return of a_2 is implicit in the period 2 state 1 budget contraint (you could write it explicity as $0 * a_2$). The Lagrangian is,

$$\begin{aligned}\mathcal{L}(c_1, c_2(z_1), c_2(z_2), a_1, a_2, \lambda_1, \lambda_2, \lambda_3) = u(c_1) + \beta[pu(c_2(z_1)) + (1-p)u(c_2(z_2))] + \lambda_1(y_1 - c_1 - a_1 - a_2) \\ + \lambda_2((1+r)a_1 + y_2(z_1) - c_2(z_1)) + \lambda_3((1+r)a_2 + y_2(z_2) - c_2(z_2))\end{aligned}$$

[35] These are essentially the 'Arrow-Debreu securities' (essentially as normally would have prices, rather than rate of return r). What matters is that we have enough assets with different returns to 'span' the space of the shocks (and time periods). An intuitive way to think about it is that we need to be able to trade between all the states independently of the others, in this model we have three states: period one, period two with shock one, and period two with shocks two. To be able to shift between all of these we need one less asset than there are states; or you can think of it as as many assets as there are states, not counting the state we are already in.

The first-order conditions of this are

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial c_1} = 0; \quad u'(c_1) + \lambda_1(-1) = 0 \\
\frac{\partial \mathcal{L}}{\partial c_2(z_1)} = 0; \quad p\beta u'(c_2(z_1)) + \lambda_2(-1) = 0 \\
\frac{\partial \mathcal{L}}{\partial c_2(z_2)} = 0; \quad (1-p)\beta u'(c_2(z_2)) + \lambda_3(-1) = 0 \\
\frac{\partial \mathcal{L}}{\partial a_1} = 0; \quad \lambda_1(-1) + \lambda_2(1+r) = 0 \\
\frac{\partial \mathcal{L}}{\partial a_2} = 0; \quad \lambda_1(-1) + \lambda_3(1+r) = 0 \\
\frac{\partial \mathcal{L}}{\partial \lambda_1} = 0; \quad c_1 + a = y_1 \\
\frac{\partial \mathcal{L}}{\partial \lambda_2} = 0; c_2(z_1) = (1+r)a_1 + y_2(z_1) \\
\frac{\partial \mathcal{L}}{\partial \lambda_3} = 0; c_2(z_2) = (1+r)a_2 + y_2(z_2)
\end{aligned}$$

From various combinations of the first five we get,

$$\begin{aligned}
u'(c_1) = (1+r)\beta p u'(c_2(z_1)) \\
u'(c_1) = (1+r)\beta(1-p)u'(c_2(z_2)) \\
pu'(c_2(z_1)) = (1-p)u'(c_2(z_2))
\end{aligned}$$

Complete markets make it possible to trade independently between any two states (we have three, period 1, period 2 shock 1, period 2 shock 2; note that, e.g., one more unit of a_1 and one less unit of a_2 'trades' between period 2 shock 1 and period 2 shock 2, with no impact on period 1). So complete markets mean we equalize the (discounted) marginal utility between any two states, adjusted for the probability of the state and for the 'price' of transfering between states ($1+r$ between period one and either of the period two states, 1 between the two period 2 states).

The most important difference to notice between the incomplete markets and complete markets models we just solved is that in the complete markets economy we had that we equate (with adjustments for prices, discounting, and probability) the marginal utility for each and every state. By contrast in the incomplete markets model it was not always possible to equate between some states, in our case the two period 2 states, and so agents are in this sense more 'exposed' to the risk of bad shocks that arise in some states; they are less able to insure against them. Because agents cannot insure so well against the shocks it follows that they are more 'affected' by the shocks and end up very different to each other based on which shocks hit them over their lifetime. The idiosyncratic shocks that hit a household in a complete markets model are largely irrelevant as households are perfectly insured agains them; so different shocks do not give rise to differences between agents in a complete markets model.

3.5.1 Representative Agent

Complete Markets are not the same thing as a Representative Agent. But complete markets are a prerequisite (a necessary condition) for the existence of a Representative Agent. We won't attempt to fully explain the concept of a Representative Agent here as it involve competitive equilibrium, while none of the life-cycle models considered in this document are general equilibrium models (see the companion document on OLG models for the extension of life-cycle models to general equilibrium, linked in Section 10).

The basic idea is that instead of solving a model with lots of different households (whether due to ex-ante differences, or simply ex-post different due to receiving different idiosyncratic shocks) we can just solve a model with a Representative Agent[36] and get the same answer. Since Representative Agent models are easier to solve, if they give the same answer we are better off just using Representative Agents. The intuition for creating a Representative Agent is that any competitive equilibrium of a model containing lots of different households can be recreated (in terms of the aggregates, like total consumption, or total assets) as the competitive equlibrium of a model with a Representative Agent as long as we choose the preferences appropriately (as a weighted sum of the individual households, with the weights based on the lagrange multipliers of their maximization problem); and as long as markets are complete.

Obviously the idea that heterogeneity does not matter is a strong assumption, and one that often fails to be realistic. Representative Agent models are 'simpler' and more parsimonious than heterogenous agent models, and for that reason are still useful for some things where either heterogeneity is unimportant, or where it would simply be impossible to solve the incomplete markets version of the model. But Representative Agents models should be treated with caution where they differ from heterogenous agent models. It is worth knowing that while you can create a Representative Agent that recreates all the same model aggregates, like consumption and assets, as would occur in a complete markets competitive equilibrium, you cannot create a Representative Agent for the welfare of these agents, and so all welfare analysis in Representative Agent models is actually not well microfounded. Just like it is still useful to teach very simple models in first-year undergraduate economics classes, it is still useful to use Representative Agent models in teaching.

Because we need the concept of competitive equilibrium to make sense of the details of Representative Agent models I won't attempt it here. A decent 'simple' example is given in 'Lecture 1: Complete Markets' by Florian Scheuer (if you know of a nicer explanation/example please let me know so I can link it here instead)

[36] Technically it is not one single agent, but a continuum of agents that are all identical, but we can solve it like there was one agent as all the other agents are identical. The continuum of agents is needed to allow that the model has a competitive equilibrium, rather than being a monopoly.

4 Further building up Life-Cycle Models

4.1 Life-Cycle model 20: Idiosyncratic shocks that depend on age

We will now look at how to set up shock that depend on age: both the grid points and the transition matrix can change with age. There are two ways to do this, either by creating matices z_grid_J and pi_z_J which contain z_grid and pi_z for each age, or by using a function that takes (age dependent) parameters as inputs and returns the grid and transition matrix for that age, called *ExogShockFn* in the codes. Life-Cycle Model A5 shows how to create z_grid_J and pi_z_J for an AR(1) process whose parameters depend on age. Here will will demonstrate how to use $ExogShockFn$.

We will use Life-Cycle Model 8, modifying it so that the transiton matrix for shocks depends on age (but the grid does not). That model has an exogenous shock z that takes two values, representing employment and unemployment, respectively. Empirically the unemployment rate is higher for the young, and job turnover is also higher for the young. We will therefore change the transition matrix so that the probability of remaining in the employment state increases in age; I have not attempted to calibrate the numbers seriously, they are purely illustrative.

We create an 'ExogShockFn' which takes in some parameters and returns z_grid and pi_z; the dependence on age comes because the parameters themselves (can) depend on age. If you look at the contents of ExogShockFn you can see how the transition matrix is created to depend on age.

The codes start all households at age $j = 1$ in the stationary distribution for the transition matrix on z at age $j = 1$. So if the transition matrix did not change with age then nor would the distribution of agents over z (the distribution of agents over a could change, but not over z). We plot the life-cycle profile showing the fraction of the population in the 'unemployment' state, and can see how it changes with age because the transition matrix on z is changing with age.

Note that the number of grid points on z must remain unchanged in age. Of course by making the probability of transitioning into some of them equal to zero you can effectively remove some.

4.2 Life-Cycle model 21: Idiosyncratic medical shocks in retirement

We just saw how to allow exogenous shock grids and transition path to depend on age. When doing this we can easily also reinterpret these exogenous shocks to have different meanings at different ages. Notice how we give significance to the values of z via the return function and the FnsToEvaluate. So if we change the return function and FnsToEvaluate to interpret z (or any other state) in a different way at different ages then the meaning of z is different for different ages. We have already seen in Life-Cycle Model 20 (and A5) how to change the grid values and transition matrix for z for different ages, so this makes it easy to completely repurpose z at different ages.

We will modify Life-Cycle Model 8, which had a shock z that takes two values representing employment and unemployment, respectively. Notice that in that model retirees never work, and so the shock was irrelevant/ignored

for all retirees. So let's repurpose the shock during retirement. We know retirees often do not run down their savings as a basic life-cycle model would predict. We have already seen one possible reason, that they want to leave bequests.[37] Another possible reason that the elderly do not run down savings is that they want substantial savings to pay for potentially large medical expenses. We will repurpose the shock z to be a medical expense shock during retirement, to create precisely this motive.[38]

The household value fn problem is,

$$\begin{aligned} V(a,z,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\ & \qquad + s_j \beta E[V(aprime, zprime, j+1)|z] \\ & \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j z h \\ & \text{if } j >= Jr: \; c + aprime = (1+r)a - z + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \\ & zprime = \pi_z(z), \text{ is a two-state markov} \end{aligned}$$

where now z also appears in the retired ($j >= Jr$) budget constraint (interpreted as substracting a medical expense).

So we are going to have the exogenous shock z represent employment/unemployment during working age, and then medical expenses during retirement. We use ExogShockFn to do this, and will just set one grid for working age and another grid for retirement; likewise we set one transition matrix for working age and another for retirement.[39]

Note that the number of grid points on z must remain unchanged in age. Of course by making the probability of transitioning into some of them equal to zero you can effectively remove some.

If you run this model and Life-Cycle Model 8, the only difference between the two is the medical shocks, and you will be able to see their impact in increased savings by the elderly in the life-cycle profiles.

4.3 Life-Cycle model 22: Deterministic Economic/productivity growth

The economy grows over time, as do incomes, as productivity rises. We will add a deterministic rate of growth of producitivity, g, at which incomes rise. The way to solve a model with deterministic growth is to look for a balanced growth path by converting the model into 'per technology unit' terms, which makes the model stationary.

[37] We modelled warm-glow bequests, but there were more a simply way to capture something like wanting to leave model for descendents (or to charity), and should be understood as a simple way to capture a complex behaviour, rather than the 'warm glow' being taken too seriously.

[38] Obviously the importance of medical expense shocks is likely to be much larger in countries with private health systems vs public health systems. The literature also points to the potentially large costs of elderly-care in nursing homes and the like as important to the savings of the elderly.

[39] We could of course set these to change with age —e.g., medical shocks become more likely as households get older— but to keep things easy to follow we will just keep the same grids and transition matrices except for the change from working age to retirement.

Then solve the stationary model, and then if wanted we can add growth back into the simulations of the stationary model.[40] Solving the stationary model can be done in the usual manner.

To be able to do this we need to use a non-seperable utility function.[41] Conceptually the reason is simple enough. If incomes grow (which is the point of our deterministic productivity growth g) then people will consume more and so the marginal utility of consumption falls. Then seperable utility implies that equating marginal utilities therefore requires a falling marginal utility of leisure, which means ever increasing leisure. So seperable utility together with growing incomes drives hours worked to decrease towards zero. With non-seperable utility the marginal utility of leisure is increasing in consumption (this is meaning of non-seperable) and so (optimal) leisure does not continually increase as consumption increases, and so hours worked to not tend to zero.

For the renormalization to work we need to be careful about what exactly grows. When using a model with exogenous labor we would have the endowment income grow at rate g, whereas when using a model with endogenous labor we need it to be the wage (per unit of labor supply/per hour worked) that grows at rate g.

A brief further discussion of what kinds of utility functions can be used with models that have growth is appropriate. You are probably familiar with the neoclassical growth model (a.k.a. Solow or Solow-Swan model). The general characterization of prefererences consistent with the existence of a steady-state in a model with deterministic producitivity growth are 'Uzawa prefereces'. Two extensions may also be of interest. The first is the extension of Uzawa preferences to models with endogneous human capital, by Grossman, Helpman, Oberfield, and Sampson (2017). The second is Boppart and Krusell (2020) which provides preferences consistent with the combination of both income growth and moderate decrease in hours worked in high income countries over the 20th century.

We will first do two models without idiosyncratic shocks. Let's start with the exogenous labor case, as it is slightly simpler, and then do the endogenous labor case. After these we solve the endogenous labor case with idiosyncratic shocks. For Assignment 4 you are provided with just the model set up for exogenous labor supply with idiosyncratic shocks and deterministic income growth; solving the equations to renormalize the the exogenous labor case is left as Assignment 4 (a code 'solution' is provided).

4.3.1 Deterministic income growth with exogenous labor supply

I will explain the method using a very simple life-cycle model to make clear the concepts, and then simply write out what Life-Cycle Model 10 becomes with deterministic growth.

[40] Notice that this is exactly the same thing as when solving the neoclassical growth model (a.k.a. Solow or Solow-Swan model); solve the balanced growth path by dividing by the technology level (and population), and then solving for the steady-state. Note that the difference between stationary equilibrium and steady-state is minor, they are often used interchangably in conversation.

[41] Non-seperable utility was discussed Assignment 2 (just after Life-Cycle model 10).

Consider the following household problem,

$$\sum_{j=1}^{J} \frac{c_j^{1-\sigma}}{1-\sigma}$$
$$c_j + a_{j+1} = (1+r)a_j + y_j$$
$$y_j = (1+g)y_{j-1}, \quad y_1 \text{ given}$$

Notice that y_j is growing at rate g; the deterministic income growth.

Trying to compute the solution to this problem in it's current form is possible (because there is a final period, there is a maximum amount that income will end up being)[42], but we would require very large grids as we would need to deal with the smallest income in period 1 and the largest income in period J, as well as the wide range of asset holdings that would result. We will therefore rewrite everything in such a way as to 'remove' the growth. The intuition is that $y_j = (1+g)^{j-1}y_1$, and so if we can just divide everything by $(1+g)^{j-1}$ then the growth would 'disappear'.

First, rewrite y_j, to get

$$\sum_{j=1}^{J} \beta^{j-1} \frac{c_j^{1-\sigma}}{1-\sigma}$$
$$c_j + a_{j+1} = (1+r)a_j + (1+g)^{j-1}y_1$$

and now divide the budget constraint through by $(1+g)^{j-1}$ to get

$$\sum_{j=1}^{J} \beta^{j-1} \frac{c_j^{1-\sigma}}{1-\sigma}$$
$$\frac{c_j}{(1+g)^{j-1}} + \frac{a_{j+1}}{(1+g)^{j-1}} = (1+r)\frac{a_j}{(1+g)^{j-1}} + y_1$$

We want this to be simpler, notice that we could 'get rid' of the denominator on c_j by defining a 'renormalized' variable $\hat{c}_j \equiv \frac{c_j}{(1+g)^{j-1}}$. We could do the same with a_j, defining $\hat{a}_j \equiv \frac{a_j}{(1+g)^{j-1}}$. We would then get

$$\sum_{j=1}^{J} \beta^{j-1} \frac{(\hat{c}_j(1+g)^{j-1})^{1-\sigma}}{1-\sigma}$$
$$\hat{c}_j + (1+g)\hat{a}_{j+1} = (1+r)\hat{a}_j + y_1$$

[42] This would not be true in an infinite horizon model, nor if we extended this to an OLG model.

notice both that we made the substitution in the utility function as well, notice also how we could substitute $\frac{a_{j+1}}{(1+g)^{j-1}}$ with $(1+g)\hat{a}_{j+1}$.

We have one more step, and for this step the functional form of the utility function becomes important (especially in the model with endogenous labor). Right now the utility function 'changes' every period (there is a j inside it), but we can pull the $(1+g)^{j-1}$ out in front to get

$$\sum_{j=1}^{J} \beta^{j-1}((1+g)^{j-1})^{1-\sigma}\frac{(\hat{c}_j)^{1-\sigma}}{1-\sigma}$$
$$\hat{c}_j + (1+g)\hat{a}_{j+1} = (1+r)\hat{a}_j + y_1$$

and with a little bit of rewriting the powers, we get

$$\sum_{j=1}^{J} \beta^{j-1}((1+g)^{1-\sigma})^{j-1}\frac{(\hat{c}_j)^{1-\sigma}}{1-\sigma}$$
$$\hat{c}_j + (1+g)\hat{a}_{j+1} = (1+r)\hat{a}_j + y_1$$

The important thing here is that now neither the utility function, $\frac{(\hat{c}_j)^{1-\sigma}}{1-\sigma}$, nor the budget constraint $\hat{c}_j + (1+g)\hat{a}_{j+1} = (1+r)\hat{a}_j + y_1$ has parameters that change with the period j (when we started y_j changed with j). In fact the deterministic growth now just appears as an additional discount factor, note that we could group the discounts factors to get, $(\beta(1+g)^{1-\sigma})^{j-1}$. So solving the model with deterministic growth, after renormalizing, just looks like a model with a different discount factor, and that is easy enough to solve.

Note that if the model had retirement and pensions, then you would either end up with the renormalized pension decreasing in j, or you could consider pensions which are indexed to the growth (if you are looking at a specific country you should follow the pension legislation).

Note, no code is provided to solve this model, you can hopefully figure it out based on the code for the model with deterministic wage growth and endogenous labor supply and idiosyncratic shocks that follows shortly.

4.3.2 Deterministic income growth with endogenous labor supply

Consider the following household problem,

$$\sum_{j=1}^{J} \beta^{j-1}\frac{c_j^{\sigma_1}l_j^{1-\sigma_1})^{1-\sigma_2}}{1-\sigma_2}$$
$$c_j + a_{j+1} = (1+r)a_j + w_j(1-l_j)$$
$$w_j = (1+g)w_{j-1}, \quad w_1 \text{ given}$$

Notice that w_j is growing at rate g; the deterministic wage growth.

There are two key differences with endogenous labor. The first is the utility function which must be non-seperable. We already discussed the intuition for this and we will see mathematically that the non-seperability is needed to be able to get the $(1+g)^{j-1}$ term out of the (renormalized) utility function so that it just becomes an additional discount factor. The second is that now we have the deterministic growth in the wage, w_j and not in the labor income $w_j(1-l_j)$[43]

From here the steps are almost exactly the same as with exogenous labor supply, first we can divide the whole budget constraint by $(1+g)^{j-1}$ and define $\hat{c}_j \equiv \frac{c_j}{(1+g)^{j-1}}$ and $\hat{a}_j \equiv \frac{a_j}{(1+g)^{j-1}}$; note that we do not change l_j, because it multiplies the w_j. Substituting we get

$$\sum_{j=1}^{J} \beta^{j-1} \frac{(\hat{c}_j(1+g)^{j-1})^{\sigma_1} l_j^{1-\sigma_1})^{1-\sigma_2}}{1-\sigma_2}$$
$$\hat{c}_j + \hat{a}_{j+1} = (1+r)\hat{a}_j + w_1(1-l_j)$$
$$w_j = (1+g)w_{j-1}, \quad w_1 \text{ given}$$

note the l_j and w_1. We can now do exactly the same step of taking the growth term out of the utility function and making it appear as just an extra discount factor,

$$\sum_{j=1}^{J} (\beta(1+g)^{\sigma_1(1-\sigma_2)})^{j-1} \frac{(\hat{c}_j)^{\sigma_1} l_j^{1-\sigma_1})^{1-\sigma_2}}{1-\sigma_2}$$
$$\hat{c}_j + \hat{a}_{j+1} = (1+r)\hat{a}_j + w_1(1-l_j)$$
$$w_j = (1+g)w_{j-1}, \quad w_1 \text{ given}$$

where the discount factor is now $\beta(1+g)^{\sigma_1(1-\sigma_2)}$.

Note, no code is provided to solve this model, you can hopefully figure it out based on the code for the model with deterministic wage growth and endogenous labor supply and idiosyncratic shocks that follows shortly.

4.3.3 Deterministic income growth with endogenous labor supply and exogenous shocks

Let's extend Life-Cycle Model 9 to include deterministic income growth, as part of which we will have to switch to a non-seperable utility function. To make things simple I am going to index pensions to the growth of wages.[44]

[43] The reason for this is more obvious in an OLG model where things will fail to add up at the aggregate (whole economy) level unless we put the deterministic growth in the wage. In the present life-cycle it is, roughly speaking, because labor supply is a decision variable that gets determined endogenously and so is not something that we can renormalize in terms of.

[44] This is not necessary, just makes the renormalized model a bit simpler.

So we start with the household problem,

$$\begin{aligned}
V(a_j, z_j, j) \quad = \quad & \max_{c_j, a_{j+1}, h_j} \frac{c_j^{\sigma_1}(1-h_j)^{1-\sigma_1})^{1-\sigma_2}}{1-\sigma_2} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(a_{j+1}) \\
& \qquad + s_j \beta E[V(a_{j+1}, z_{j+1]}, j+1)|z] \\
& \text{if } j < Jr: \; c_j + a_{j+1} = (1+r)a_j + w_j \kappa_j z_j h_j \\
& \text{if } j >= Jr: \; c_j + a_{j+1} = (1+r)a_j + pension_j \\
& 0 \leq h \leq 1, a_{j+1} \geq 0 \\
& log(z_j) = \rho_z log(z_{j-1}) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \\
& w_j = (1+g)w_{j-1}, \quad w_1 \text{ given} \\
& pension_j = (1+g)pension_{j-1}, \quad pension_{Jr} \text{ given}
\end{aligned}$$

notice that both wage, w_j, and pension, $pension_j$ grow at deterministic rate g (from w_1 and $pension_{Jr}$, respectively). Note that I have had to change notation slightly from Life-Cycle Model 9, in the sense that we use c_j rather than c, and similarly for the other variables, purely because it makes the renormalization look more like those we have already seen.

We can do the same renormalization as before, defining $\hat{c}_j \equiv \frac{c_j}{(1+g)^{j-1}}$ and $\hat{a}_j \equiv \frac{a_j}{(1+g)^{j-1}}$, and leaving h_j and z_j unchanged. We can substitute these in and then extract the term on the utility function to be an additional discount factor to get,

$$\begin{aligned}
V(\hat{a}_j, z_j, j) \quad = \quad & \max_{\hat{c}_j, \hat{a}_{j+1}, h_j} \frac{\hat{c}_j^{\sigma_1}(1-h_j)^{1-\sigma_1})^{1-\sigma_2}}{1-\sigma_2} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(\hat{a}_{j+1}(1+g)^j) \\
& \qquad + s_j \beta E[V(\hat{a}_{j+1}, z_{j+1]}, j+1)|z] \\
& \text{if } j < Jr: \; \hat{c}_j + (1+g)\hat{a}_{j+1} = (1+r)\hat{a}_j + w_1 \kappa_j z_j h_j \\
& \text{if } j >= Jr: \; \hat{c}_j + (1+g)\hat{a}_{j+1} = (1+r)\hat{a}_j + pension_{Jr} \\
& 0 \leq h \leq 1, \hat{a}_{j+1} \geq 0 \\
& log(z_j) = \rho_z log(z_{j-1}) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon,z})
\end{aligned}$$

note that we now just have w_1 and $pension_{Jr}$. Notice also the $warmglow(\hat{a}_{j+1}(1+g)^j)$; we could just take the $(1+g)^j$ out of the warmglow function as really it is just equivalent changing two of the warm-glow parameters, $warmglow1$ and $warmglow2$. When coding we will take advantage of this to rewrite the warm-glow as $(1-s_j)\beta(1+g)^{-1}\mathbb{I}_{(j>=Jr+10)} warmglow(\hat{a}_{j+1}(1+g))$, as mentioned this is really just reparametrizing, not changing anything of substance.

We will solve this model in the standard way, and simulate some panel data with it. Note that what we simulate will be $\hat{a}$, $w_1\kappa_j z_j h$, and h. We will then create panel data for the original model, by changing these to a, $w_j\kappa_j z_j h$ and h; this requires multiplying the first two by $(1+g)^{j-1}$ but leaving the third unchanged (we need to modify to get a and w_j).

4.3.4 Assignment 4: Deterministic income growth with exogenous labor supply and exogenous shocks

Life-Cycle Model 10, with deterministic growth in the stochastic endowment is given by,

$$\begin{aligned}
V(a,z,j) &= \max_{c,aprime} \frac{c^{1-\sigma}}{1-\sigma} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\
&\qquad +s_j\beta E[V(aprime, zprime, j+1)|z] \\
&\text{if } j<Jr: \ c+aprime = (1+r)a + w_j kappa_j z \\
&\text{if } j>=Jr: \ c+aprime = (1+r)a + pension_j \\
&aprime \geq 0 \\
&log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \\
&w_j = (1+g)w_{j-1}, \quad w_1 \text{ given} \\
&pension_j = (1+g)pension_{j-1}, \quad pension_{Jr} \text{ given}
\end{aligned}$$

for Assignment 4 you first need to convert this model into one where there is a stationary equilibrium (by normalizing by the income growth), and then solve the model with code. Then use the solution to the stationary model to simulate some panel data, and then put the deterministic income growth back into the panel data.

We mentioned previously that with exogenous labor supply it is income that grows, while with endogenous labor it is important that it is the wage that grows. So how come we have the growth in the 'wage' in this exogenous growth model? Because a 'wage' is somewhat meaningless for exogenous labor, and notice that there is no difference between $w_j kappa_j z$ growing and w_j growing if they are all constants (or stationary stochastic processes).

A solution code for Assignment 4 is provided, but try to solve without looking at it.

4.4 Life-Cycle model 24: Permanent Types: Solving fixed-types

We will use permanent types, which is a way to solve N_i household models at once. Our first model is just that of Life-Cycle model 11, with endogenous labor and both persistant and transitory shocks to labor efficiency units. In this model we add a 'fixed effect' to the labor efficiency units, denoted α_i, something common in the literature. This is the only change to the model. Permanent types, or PTypes, is how VFI Toolkit handles different types of

agents, it is much more general than just fixed-types as will be seen in later Life-Cycle models, but for now we simply use it for this purpose.

We add a parameter, α_i, which depends on the permanent types indexed by i. We will use N_i different permanent types of agents, and simply by setting the parameter α_i to have N_i different values the toolkit will automatically recognise that this parameter depends on the permanent type and respond appropriately.

Our household value function now includes a fixed-effect α_i to determine idiosyncratic producitivity units,

$$\begin{aligned} V_i(a,z,e,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta\mathbb{I}_{(j>=Jr+10)}warmglow(aprime) \\ & +s_j\beta E[V_i(aprime,zprime,eprime,j+1)|zprime] \\ & \text{if } j<Jr: \ c+aprime=(1+r)a+w\kappa_j\alpha_i zeh \\ & \text{if } j>=Jr: \ c+aprime=(1+r)a+pension \\ & 0\leq h\leq 1, aprime\geq 0 \\ & log(zprime)=\rho_z log(z)+\epsilon, \ \epsilon\sim N(0,\sigma^2_{\epsilon,z}) \\ & log(e)\sim N(0,\sigma^2_e) \end{aligned}$$

Notice that z is AR(1) and e is i.i.d. normal. Note that the codes use 'e' variables for i.i.d. shocks, see Life-Cycle model 11 if you have not previously seen this feature.

Notice that the value function now depends on i: V_i. So we are now essentially solving N_i seperate problems.

In the codes we will use $N\_i = 5$. We just set parameter α_i to be a vector of length N_i and the toolkit will automatically realise that this parameter depends on the permanent type and act appropriately. The other noteworthy change is that we have to switch all the commands to *_PType*, e.g., *ValueFnIter_Case1_FHorz_PType* instead of *ValueFnIter_Case1_FHorz*, and similarly for all commands around the agent distribution, function evaluation, etc. Of course the 'shape' of the results changes: V now becomes a structure, with *V.ptype*001,..., *V.ptype*005 being the value functions for each of the five permanent types.[45] Similarly thinks like the agent distribution will now have one distribution for each permanent type. When we calculate something like life-cycle profiles the output now provides both a result conditional on each permanent type, and a grouped result.

While it is not necessary for solving the value function problem we do need to state the weights of each of the different permanent types to solve the stationary distribution. This is a vector-parameter which we will call *alphadist*, and we need to put the name of this in *PTypeDistParamNames*, which we then pass to the stationary distribution commands. This information gets encoded into the StationaryDist, and so we do not need to include it seperately later when doing things like life-cycle profiles.

This example plots life-cycle profiles for α_i, both for each permanent type, and grouped across the permanent types. This provides a nice example to show how the toolkit has interpreted α_i is a parameter that differs by

[45] *ptype*001, etc., are the default 'names' given to the fixed-types, as later Life-Cycle model examples using permanent types will make clear we can actually choose our own names.

permanent type, and how the grouped value is simply the values of α_i summed according to their weights in *alphadist*.

4.5 Life-Cycle model 25: Using Names for Permanent Types: patient and impatient

We consider a model with two agents who differ by the value of their discount factor parameter, β. We can use names for the agents, and so call them 'patient' and 'impatient'. The model is just that of Life-Cycle Model 11, except that there are now two agents with different values of β_i, indexed by i. The main purpose of this example is to show how we can use names for permanent types of agents in codes.

Our household value function now includes two permanent types that differ by their value of β_i, the discount factor parameter.

$$
\begin{aligned}
V_i(a,z,e,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta_i \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\
& \qquad +s_j\beta E[V_i(aprime, zprime, eprime, j+1)|zprime] \\
& \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j zeh \\
& \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\
& 0 \leq h \leq 1, aprime \geq 0 \\
& log(zprime) = \rho_z log(z) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \\
& log(e) \sim N(0, \sigma^2_e)
\end{aligned}
$$

Notice that z is AR(1) and e is i.i.d. normal. Note that the codes use 'e' variables for i.i.d. shocks, see Life-Cycle model 11 if you have not previously seen this feature.

Rather than use $N\_i = 2$, like we did in Life-Cycle model 24, we can instead give the agents names using $Names\_i = \{'patient', 'impatient'\}$. The PType commands will automatically use these names for all outputs. We set the different parameter values as $Params.beta.patient = 0.96$ and $Params.beta.impatient = 0.9$. After this, other than using $Names\_i$ everywhere we had N_i in Life-Cycle model 24, there is no other change we need to make.

The names we give the permanent types are automatically used for output, hence we get *V.patient* and *V.impatient* as the two value functions for the agent types. Commands for things like life-cycle profiles give us both the two life-cycle profiles conditional on each agent type (e.g., $AgeConditionalStats.earnings.patient.Mean$ and $AgeConditionalStats.earnings.impatient.Mean$) and the life-cycle profiles for the whole population (e.g., $AgeConditionalStats.earnings.Mean$).

4.6 Life-Cycle model 26: More Permanent Types

There are lots of other things you can do with permanent types: different return functions, different number of periods, different exogenous shock processes, and much much more. Hopefully more examples coming soon :D

If there is a particular thing you think would be good to see, please email or use forum: discourse.vfitoolkit.com.

The following Life-Cycle model 27 also uses permanent types, for the same purpose of modelling a fixed-effect as was seen in Life-Cycle model 24.

4.7 Life-Cycle model 28: Two decision variables (dual-earner household)

We solve a model of a household in which there are two people (e.g., a married household). They make a joint decision about how much each will work. This involves two decisions variables for the two labor supply choices. We will set up each household member to have effective hours shocks that are a combination of one AR(1) persistent shock and one i.i.d. shock (as in Life-Cycle model 11), for each person (so two of each for the household). We will also allow for correlation between the persistent shocks of the two household members, and for correlation between the transitory shocks of the two household members. We will also allow different deterministic age-dependent labor efficiency units, κ_j, for each spouse.

Our household value function now has $z1$ and $e1$ for the first spouse, and $z2$ and $e2$ for the second spouse. We also have $\kappa_{j,1}$ and $\kappa_{j,2}$. We add a disutility term for each spouses labor supply, but the household has joint consumption.

$$
\begin{aligned}
V(a,z_1,z_2,e_1,e_2,j) \quad = \quad & \max_{c,aprime,h_1,h_2} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h_1^{1+\eta}}{1+\eta} - \psi\frac{h_2^{1+\eta}}{1+\eta} + (1-s_j)\beta\mathbb{I}_{(j>=Jr+10)}warmglow(aprime) \\
& \qquad +s_j\beta E[V(aprime,z_1prime,z_2prime,e_1prime,e_2prime,j+1)|z_1,z_2] \\
& \text{if } j<Jr: \; c+aprime=(1+r)a+w\kappa_{j,1}z_1e_1h_1+w\kappa_{j,2}z_2e_2h_2 \\
& \text{if } j>=Jr: \; c+aprime=(1+r)a+pension \\
& 0\leq h\leq 1, aprime\geq 0 \\
& log([z_1prime;z_2prime])=\rho_z log([z_1;z_2])+\epsilon, \; \epsilon\sim N(0,\Sigma_{\epsilon,z}^2) \\
& log([e_1,e_2)\sim N(0,\Sigma_e^2)
\end{aligned}
$$

Notice that $[z_1, z_2]$ is VAR(1) (in logs) and $[e_1, e_2]$ is i.i.d. normal (in logs).

We are allowing for z_1 and z_2 to follow a VAR(1) (so both ρ_z and $\Sigma_{\epsilon,z}$ are 2-by-2 matrices). We do not require that $\Sigma_{\epsilon,z}$ be diagonal, so the innovations themselves can be correlated. We similarly allow for e_1 and e_2 to be correlated (so Σ_e is not diagonal).

Correlated shocks, which use jointly-determined grids, are explained in Life-Cycle model A9.

This model demonstrates how to use two decision variables, you need to put them as the first two entries to the ReturnFn and also to all FnsToEvaluate. The model output includes calculating the life-cycle labor supply

for each spouse seperately, as well as the total household labor supply.

A common variation of the model used in practice would add a fixed cost of working (e.g., $-\mathbb{I}_{h_2>0} * fc$, where fc is a constant) to capture that the second spouse in many households will sometimes supply zero labor.[46] Note that implementing this is just a simple modification of the ReturnFn.

Because the household has two labor supply decisions the two spouses are able to insure each other against poor labor market outcomes. This is known as 'intra-household' insurance. If you are interested in this kind of model, see Ortigueira and Siassi (2013).

4.8 Life-Cycle model 29: Semi-exogenous state (fertility and children)

We will modify Life-Cycle model 9 to include a fertility decision and children. This involves adding one decision variable, the fertility, and two semi-exogenous states, the number of infants and the number of children. A semi-exogenous state is an exogenous state the transition probabilities of which depend on a decision variable.[47] In our case, the transitions around how many infants the household has will depend on the fertility decision.

The modifications to the state variables from Life-cycle model 9 are thus to add a fertility decision, f, which takes the values of zero or one. A semi-exogenous state n_1, which is the number of infants and takes the values of zero or one. A second semi-exogenous state n_2, which is the number of children and can take the values 0,1,2,3.[48] Let's see the household problem, and after that we will discuss the semi-exogeneity.

Households have children because they like them, that is they get utility directly from the presence of children. We use the functional form $\eta_1 \frac{exp(j-\eta_3)}{(1+exp(j-\eta_3))}(\bar{n}+n)^{\eta_2}$, notice that this depends on age and $\bar{n}$ acts as the desired number of children. Models of this kind often include things about the time required to look after the infant and the cost of childcare if working, and these just involve small changes to the return function.

[46] Because we have $0 \leq h_2 \leq 1$ together with the shape of the utility function it would never be optimal to set $h_2 = 0$ in the absence of a fixed cost of working non-zero hours.

[47] Not to be confused with a semi-endogenous state, which is an exogenous state the transitions of which depend on an endogenous state. The difference is dependence on an endogenous state versus dependence on a decision variable.

[48] This could go higher than 3. When we define the actual transition probabilities you will see that n_2 is actually exogenous, but because it will depend on n_1 and because n_1 is semi-exogenous it 'inherits' this semi-exogeneity from the perspective of the codes.

Our household value function is now,

$$
\begin{aligned}
V(a,n_1,n_2,z,j) \quad = \quad & \max_{c,aprime,h,f} \frac{(c/equiv)^{1-\sigma}}{1-\sigma} - \psi\frac{(h+infantcaretime)^{1+\eta}}{1+\eta} + \eta_1\frac{exp(j-\eta_3)}{(1+exp(j-\eta_3))}(\bar{n}+n)^{\eta_2} + \\
& (1-s_j)\beta\mathbb{I}_{(j>=Jr+10)}warmglow(aprime) \\
& +s_j\beta E[V(aprime,n_1prime,n_2prime,zprime,j+1)|z] \\
& \text{if } j<Jr: \ c+aprime=(1+r)a+w\kappa_j zh-childcarecosts \\
& \text{if } j>=Jr: \ c+aprime=(1+r)a+pension \\
& equiv=equiv_1n_1+equiv_2n_2 \\
& infantcaretime=h_cn_1 \\
& childcarecosts=childcarec(h>0)n_1 \\
& 0\leq h\leq 1, aprime\geq 0 \\
& log(zprime)=\rho_z log(z)+\epsilon, \ \epsilon\sim N(0,\sigma^2_{\epsilon,z})
\end{aligned}
$$

where *equiv* is a household consumption equivalence scale (scales down the utility of consumption based on the number of people in the households). Notice that n_1 and n_2 are part of the state space, and f is added to the decision variables. Infants require the household to decicate some of their time ($infantcaretime$), and if they work ($h>0$) while they have an infant there are child care costs ($childcarecosts$).

We first discuss semi-exogeneity in terms of this application, and then give a more general mathematical description. We have two semi-exogenous variables n_1 and n_2 whose transition probabilities depend on the fertility decision f. The idea is that n_1 is the number of infants, and if it is currently zero later period and the household is not trying to have a child —the fertility decision f equals zero— then the number of infants next period is zero. If however the number of infants is zero and the fertility decision equals one, then the household will succeed in having an infant next period with probability $probofbirth_j$ (and remain with zero infants next period with probability $1-probofbirth_j$). Hence the transition probabilities for the semi-exogenous state n_1 depend on the value of the decision variable of fertility. Notice that the probability of birth depends on age j. The rest of relatively exogenous and simply inherits the semi-exogeneity from the part we just described: if the household already has an infant, $n_1=1$, then that infant becomes a child (adding one to n_2, and subtracting one from n_1) with probability $probofchild$. Children, n_2, become adults with probability $probofadult$, which has the effect of substracting one from n_2.

To set up the semiexogenous variables in the code we have a few steps: (i) define in $vfoptions$ the fields n_semiz, $semiz\_grid$, and $SemiExoStateFn$, setting up the first two of these is fairly standard (ii) set up $SemiExoStateFn$, which takes the inputs of $(semiz, semizprime, d, ...)$ and returns the transition probability for the next period $semizprime$ given this period $semiz$ and the decision d as well as any parameters. We only tell the codes about the semi-exogenous states, here n_1 and n_2, but not explicitly about the relevant decision variable, here f. VFI

Toolkit is hard-coded to assume that the 'last' decision variable is the one that is relevant to the semi-exogenous states (if these are being used in $vfoptions$). (iii) We also need to set up the same three fields in $simoptions$. (iv) The ordering of the inputs to the return function are: $(d, aprime, a, semiz, z, e, ...)$, although of course the current problem does not use any e variables, I just mention them to be comprehensive.[49]

From the mathematical perspective when happends is VFI Toolkit treats $semiz$ variables the same as z variables for things like evaluating the return function, but then for computing expectations or the agent distribution it creates a large transition matrix for the $semiz$ corresponding to each possible value of the 'last' decision variable. Hence from the perspective of solving models, any situation in which you have $Pr(semiz'|semiz, d)$ can be solved using VFI Toolkit with a semi-exogenous shock. That is, any time you have a 'exogenous variable whose transition probabilities depend on a decision variable'. The key limitations are that the next period value of the semi-exogenous variable cannot appear in the return function, and that it is a decision variable (not an endogenous state variable) on which the transition probabilities depend.[50]

At the bottom of the codes after the model is solved it shows what the transition matrix that is created internally, with transition probabilities from $semiz$ to $semizprime$ which depend on d, look like. This might help you understand exactly what it is and is not capable of.

5 Solving Models Faster!

5.1 Life-Cycle model 30: Exploiting Monotonicity for Faster Solutions (Divide-and-Conquer)

No actual model. Instead just add a single line of code to any of the Life-Cycle Models we have seen so far, namely *vfoptions.divideandconquer=1* and as a result, it solves more than twice as fast! In fact the model also uses less GPU memory when solving, meaning it is not only faster, but that you can solve bigger models on a given GPU![51]

Divide-and-conquer works by exploiting the monotonicity of the policy function, that is, it is exploits that $aprime(a)$ is a monontone increasing function (that next periods choice of endogenous state is a monotone increasing choice of this periods endogenous state). This is a property that almost all Economic models satisfy, although there are some expections.

In practice, you would want to set *vfoptions.divideandconquer=1* in every single one of the models we have seen. These models all satisfy the monotonicity assumption, and we can solve them faster by exploiting this property. The reason that using divide-and-conquer is not on by default is there are some non-monotone models, and so it is important that you think about whether your model is monotone before you activate *vfoptions.divideandconquer=1*.

[49] The ordering of inputs for any functions to evaluate are the same, $(d, aprime, a, semiz, z, e, ...)$.

[50] For example, another application would be an employed/unemployed/out-of-labor-force semi-exogenous state, where the transition probabilties depend both of the previous value of this semi-exogenous state and on a decision variable about how hard to search for a job.

[51] Because of how GPUs parallelize there is an exception, namely for very small models it may actually be slower.

Divide-and-conquer in VFI Toolkit works in two 'layers'. *vfoptions.level1n=11* is the number of points (in the grid on this period endogenous state) used in the first layer. If you increase/decrease this number you can make the codes go faster still (the best number depends heavily on the model, so you just have to play around with it (the speed gains are modest compared to turning on divide-and-conquer, so you can just leave it at default value unless you really do need the speed).

If you are unsure if your model satisfies the monotonicity assumption, you have two options: (i) sit down and think carefully through the equations, (ii) run the model twice with *vfoptions.divideandconquer=0* and *vfoptions.divideandconquer=1* and check that both give identical answers (which they only will if the model is monotone; if the answers differ the model is non-monotone and only the *vfoptions.divideandconquer=0* answer is correct). Obviously the first of these is the more through and better, but the second is very easy and convenient (albeit not completely certain).

For a more comprehensive explanation of what divide-and-conquer is and what exactly the technical monotonicity assumption involves, see: http://discourse.vfitoolkit.com/t/divide-and-conquer-solve-one-endogenous-state-models-faster/265/39

6 Portfolio-Choice in Life-Cycle Models (riskyasset)

6.1 Life-Cycle model 31: Portfolio-Choice

Households saving for retirement face a 'portfolio-choice' decision about how to allocate their savings; households can either save in a safe asset that returns r with certainty, or in a risky-asset that has a stochastic return (and a higher expected return). We build on Life-Cycle model 9, adding the decision on whether to invest in the safe or risky asset (and switching to exogenous labor for simplicity). We take advantage of the insight that (in the absence of portfolio-adjustment costs) once we have the returns to each asset we only need to know total assets next period. This way we can have only one endogenous state (total assets), and then need two decision variables, savings and the 'share of savings invested in the risky asset'. Note however that this means we cannot choose next period endogenous state (total assets) directly, as it depends on our two decision variables, but also on the return to the risky asset which depends on an i.i.d. shock that occurs between this period and next.

Because *aprime* can no longer be choosen directly, we cannot use a standard endogenous state as we have done for all the examples until now. Instead we will use a '*riskyasset*', which is where $aprime(d,u)$, that is the next period endogenous state, *aprime*, is a function of the decision variables d and an i.i.d. shock u that occurs between this period and next period. Until now the return function and any functions to evaluate always had $(d, aprime, a, z...)$ as their first arguments, now with a *riskyasset* they will instead have $(d, a, z, ...)$ as their first arguments. We also need to set up $aprime(d,u)$, which will be a function that takes d and u as inputs and returns the value of *aprime* (the codes will then interpolate this onto the grid on the endogenous state).

Our household value function is thus,

$$
\begin{aligned}
V(a,z,j) \quad = \quad & \max_{c,savings,riskyshare} \frac{c^{1-\sigma}}{1-\sigma} + s_j \beta E[V(aprime, zprime, j+1)|z] \\
& \text{if } j < Jr: \ c + savings = a + w\kappa_j z \\
& \text{if } j >= Jr: \ c + savings = a + pension \\
& \quad aprime = (1+r)(1-riskyshare)savings + (1+r+u)\, riskyshare\, savings \\
& 0 \leq riskyshare \leq 1, \quad savings \geq 0 \\
& log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \\
& u \sim N(rp, \sigma^2_u)
\end{aligned}
$$

where r is the return on safe assets, and u is the i.i.d. excess return on risky assets which has mean rp (the risk premium that will be greater than zero to compensate for the risk).

Implementing the codes is almost all just the same as usual. We need to set up the $aprime(d,u)$ function (which outputs the value of *aprime*). Then when calling the value function, agent simulation, etc., we just have to set *vfoptions.riskyasset=1*.

Everything else is standard, except of course that the policy function, *Policy*, is slightly different than before as it now just contains the decision variables d.

To make things run faster we will also use *vfoptions.refine_d*, which is always a 1-by-3 vector. The three elements are respectively the number of *d1*, *d2* and *d3* decision variables, where *d1* are decision variables that appear in the return function but not the aprime function, *d2* are decision variables that appear in the aprime function but not the return function, and *d3* are decision variables that appear in both the aprime function and the return function. As a result the first inputs to the return function are $(d1, d3, a, z, ...)$, while the first inputs to the aprime function are $(d2, d3, u, ...)$. Our current model has two decisions, riskyshare and savings: riskyshare is only needed in the aprime function so it is a *d2* variable, and savings is needed for both the return function and the aprime function so it is a *d3* variable, and this model has no *d1* variables. Hence we set *vfoptions.refine_d*$= [0, 1, 1]$, and we have to make sure that the ordering used for the decision variables in *n_d* and *d_grid* fit with the same ordering (we don't have a $d1$ in this model, so first is our $d2$ which is riskyshare, and then comes our $d3$ which is savings). Note, the codes allow for more than one of each of $d1$, $d2$ and $d3$ variables; you can also have no $d1$ variables, but you cannot have no $d2$ nor no $d3$ variables. Note, functions to evaluate still take all decision variables. You also need to set *simoptions.refine_d=vfoptions.refine_d*.

6.2 Life-Cycle model 32: Portfolio-Choice with Epstein-Zin preferences

When saving for retirement and making portfolio-choice decisions households need to distinguish two things: how much to save for retirement, which will depend on the intertemporal elasticity of substitution, and how what share of savings to invest in risky assets, which will depend on the level of risk aversion. But in standard (vonNeumann-Morgenstern) preferences, these are both determined by the same parameter. Epstein-Zin preferences introduce an additional parameter so that the intertemporal elasticity of substitution and the risk aversion can be determined seperately. We now resolve Life-Cycle model 31 but this time using Epstein-Zin preferences. We will use Epstein-Zin preferences in utility-units (see Appendix B.1).

Our household value function is thus,

$$
\begin{aligned}
V(a,z,j) \quad = \quad & \max_{c,savings,riskyshare} \frac{c^{1-\sigma}}{1-\sigma} - \beta E[-s_j V(aprime, zprime, j+1)^{1+\phi}|z]^{\frac{1}{1+\phi}} \\
& \text{if } j < Jr: \ c + savings = a + w\kappa_j z \\
& \text{if } j >= Jr: \ c + savings = a + pension \\
& \quad aprime = (1+r)(1-riskyshare)savings + (1+u)\, riskyshare\, savings \\
& 0 \leq riskyshare \leq 1, \quad savings \geq 0 \\
& log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \\
& u \sim N(rp, \sigma^2_u)
\end{aligned}
$$

where the only differences relate to the preferences. Here $\phi > 0$ is a parameter that determines the amount of 'additional' risk aversion (relative to the standard vonNeumann-Morgenstern risk preferences).

Implementing this in the codes is just a few lines which say that we want to use Epstein-Zin preferences (*vfoptions.exoticpreferences='EpsteinZin';*), that the utility function is negative valued (*vfoptions.EZpositiveutility=0*), and then stating which parameter is the one relating to Epstein-Zin preferences (*vfoptions.EZriskaversion='phi'*). For more about Epstein-Zin preferences see the Appendix B.1.

When using Epstein-Zin preferences conditional surivival probabilities have to be treated specially. We therefore have to specify the name of the parameter,

vfoptions.survivalprobability='sj'

and we also remove s_j from the discount factors (which is how we treated conditional survival probabilities under standard vonNeumann-Morgenstern risk preferences).

6.3 Life-Cycle model 33: Portfolio-Choice with Warm-Glow of Bequests

With Epstein-Zin preferences there is an important difference between the discount factor and the conditional survival probability that we have to take into account when using Warm-Glow of Bequests. We add a Warm-Glow of Bequests to Life-Cycle model 32 and show how to ensure that this is handled correctly in the presence of Epstein-Zin preferences.

Our household problem is,

$$
\begin{aligned}
V(a,z,j) \quad = \quad & \max_{c,savings,riskyshare} \frac{c^{1-\sigma}}{1-\sigma} - \beta E[-s_j V(aprime, zprime, j+1)^{1+\phi} - (1-s_j)W(aprime)|z]^{\frac{1}{1+\phi}} \\
& \text{if } j < Jr: \; c + savings = a + w\kappa_j z \\
& \text{if } j >= Jr: \; c + savings = a + pension \\
& \quad aprime = (1+r)(1-riskyshare)savings + (1+u)\, riskyshare\, savings \\
& 0 \leq riskyshare \leq 1, \quad savings \geq 0 \\
& log(zprime) = \rho_z log(z) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \\
& u \sim N(rp, \sigma^2_u)
\end{aligned}
$$

where $W(aprime)$ is the warm-glow of bequest function.

The question is then just what we want to use as the warm-glow of bequest function. The most obvious choice is the utility function, and thus we get,

vfoptions.WarmGlowBequestsFn$= wg * \frac{aprime^{1-\sigma}}{1-\sigma}$

Notice that this is utility function in terms of *aprime*, but also has a parameter wg multiplying it that we can use to control the strength of the bequest motive. Any alternative function could be used for the warm-glow of bequests, which has as inputs the next period endogenous state (*aprime*) and any parameters.

Because we already had so specify the conditional surivival probabilities when using Epstein-Zin preferences this warm-glow of bequest function is simply evaluated whenever there is a non-zero risk of death (when the conditional surivival probability is less than one).

6.4 Life-Cycle model 34: Portfolio-Choice with Endogenous Labor Supply

We will now add endogenous labor supply to Life-Cycle model 33. This just involves adding a decision variable.

Our household problem is,

$$
\begin{aligned}
V(a,z,j) \quad = \quad & \max_{c,savings,riskyshare,h} \frac{c^{1-\sigma}}{1-\sigma} + \psi \frac{(1-h)^{1+\eta}}{1+\eta} - \beta E[-s_j V(aprime, zprime, j+1)^{1+\phi} - (1-s_j) W(aprime)|z]^{\frac{1}{1+\phi}} \\
& \text{if } j < Jr: \; c + savings = a + w\kappa_j z h \\
& \text{if } j >= Jr: \; c + savings = a + pension \\
& \quad aprime = (1+r)(1-riskyshare) savings + (1+u)\, riskyshare\, savings \\
& 0 \leq riskyshare \leq 1, \quad savings \geq 0 \\
& log(zprime) = \rho_z log(z) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \\
& u \sim N(rp, \sigma^2_u)
\end{aligned}
$$

where $W(aprime)$ is the warm-glow of bequest function.

Note that if we were using Epstein-Zin preferences in consumption-units we would not be able to have a utility function, as here, which is seperable in consumption and leisure. With Epstein-Zin preferences in utility-units this is not a problem.

Remember how we go about setting up *vfoptions.refine_d*. The three elements are respectively the number of *d1*, *d2* and *d3* decision variables, where *d1* are decision variables that appear in the return function but not the aprime function, *d2* are decision variables that appear in the aprime function but not the return function, and *d3* are decision variables that appear in both the aprime function and the return function. As a result the first inputs to the return function are $(d1, d3, a, z, ...)$, while the first inputs to the aprime function are $(d2, d3, u, ...)$. Our current model has three decisions, h, riskyshare and savings: h is only needed in the return function so it is a *d1* variable, riskyshare is only needed in the aprime function so it is a *d2* variable, and savings is needed for both the return function and the aprime function so it is a *d3* variable, and this model has no *d1* variables. Hence we set *vfoptions.refine_d*$= [1, 1, 1]$, and we have to make sure that the ordering used for the decision variables in *n_d* and *d_grid* fit with the same ordering (so h, riskyshare, savings). Note, the codes allow for more than one of each of $d1$, $d2$ and $d3$ variables; you can also have no $d1$ variables, but you cannot have no $d2$ nor no $d3$ variables. Note, functions to evaluate still take all decision variables. You also need to set *simoptions.refine_d=vfoptions.refine_d*.

6.5 Life-Cycle model 35: Portfolio-Choice with Housing

We will now add housing to Life-Cycle model 32. This involves adding a standard endogenous state alongside our 'riskyasset' state. We will make housing take six values, the smalllest of which is zero and represents not owning a house.

Households get utility from consumption, c and housing services s. Homeowners get housing services as a fraction of their house value/size, renters get housing services equal to half the housing services of owning the smallest house size and have to pay rent. When households buy/sell a house there are housing transaction costs, *htc*. Previously we imposed that $savings \geq 0$ (implicitly via the grid on savings), now we allow borrowing up to a fraction, f_coll, of the house value (using the house as collateral); when borrowing the borrowed assets are restricted to be safe assets (a mortgage).

Our household problem is,

$$
\begin{aligned}
V(h,a,z,j) \quad = \quad & \max_{c,savings,riskyshare,hprime} \frac{(c^{1-\sigma_h}s^{\sigma_h})^{1-\sigma}}{1-\sigma} - \beta E[-s_j V(hprime, aprime, zprime, j+1)^{1+\phi}|z]^{\frac{1}{1+\phi}} \\
& \text{if } j < Jr: \; c + savings = a + w\kappa_j z + (h - hprime) - rentalcosts - htc \\
& \text{if } j >= Jr: \; c + savings = a + pension + (h - hprime) - rentalcosts - htc \\
& \text{if } h > 0: \; s = housingservices * h, \, rentalcosts = 0 \\
& \text{if } h = 0: \; s = 0.5 * housingservices * minhouse \, rentalcosts = rentalprice \\
& \text{if } hprime = h: \; htc = 0 \\
& \text{if } hprime \neq h : htc = f\_htc(h + hprime) \\
& \quad aprime = (1+r)(1 - riskyshare)savings + (1+u)\, riskyshare\, savings \\
& 0 \leq riskyshare \leq 1 \\
& savings \geq -f\_coll * hprime \\
& \text{if } savings < 0: \; riskyshare = 0 \\
& \text{if } j >= Jr: \; savings \geq 0 \\
& log(zprime) = \rho_z log(z) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \\
& u \sim N(rp, \sigma^2_u)
\end{aligned}
$$

Notice that utility is now based on $c^{1-\sigma_h}s^{\sigma_h}$, which is an aggregate of consumption and housing services s. *minhouse* is the smallest (non-zero) house size/value.

There is also a totally arbitrary restriction that assets (savings) have to be greater than zero during retirement. Not a lot of thought has been put into the constraints of this model, they are rather arbitrary and intended as an illustration.

Because of how the toolkit works, the riskyasset must be the 'last' (here second) endogenous state.

7 Behavioural Life-Cycle Models

We now turn to various behavioural models. We look at behavioural models for modelling the concepts of impatience (Quasi-Hyperbolic Discounting), loss-aversion (Prospect Theory), temptation and self-control (Gul-Pesendorfer Preferences), unknown unknowns (Ambiguity Aversion). We also already saw Epstein-Zin preferences in Life-Cycle Model 12, which seperate risk aversion from the intertemporal elasticity of substitution. Most of these are based off of Life-Cycle Model 10 (exogenous labor and an idiosyncratic markov shock; here without warm-glow of bequests[52]), and modify it for a variety of behavioural aspects (the exception is Ambiguity Averion which is based off of Life-Cycle Model 21). For impatience and tempation we also have a 'Model B', which has endogenous labor and is based off of Life-Cycle Model 9 (except without warm-glow of bequests).

7.1 Life-Cycle model 36: Impatience (Quasi-Hyperbolic Discounting)

Quasi-Hyperbolic discounting is a way to model 'impatience'. Impatient households are those that take decisions today, which put little weight on the future, and which their future-self would not like. Every model until now has used 'exponential discounting', with every future period discounted by factor β. The idea of quasi-hyperbolic discounting is that while you still use β to discount between any two future periods, you use an additional discount factor β_0 to discount between the current period and next period; so you take decisions today that put little weight on the future.

There are two types of quasi-hyperbolic discounting, called *sophisticated* and *naive*, based on whether are sophisticated and recognise that your future-self will suffer the same impatience problems as you do, or whether you are naive and simply (incorrectly) assume your future-self will not suffer from the same impatience, respectively.

We will use Life-Cycle Model 10, and simply change to quasi-hyperbolic discounting. This is easy to code as we essentially just tell $vfoptions$ to use quasi-hyperbolic discounting and add the 'additional' discount factor. Near the start of the codes we choose which of naive and sophisticated quasi-hyperbolic discounting to use. Notice that quasi-hyperbolic discounting affects how the household makes decisions are therefore needs to be in $vfoptions$, but it is irrelavant once the policy function is known, and therefore is not in $simoptions$.

We start with the naive quasi-hyperbolic discounting problem. Notice first that the naive quasi-hyperbolic discounter (naively) believes their future self will act like an exponential discounter. So we first define the 'con-

[52] We would have to be careful about how the warm-glow of bequests might interact with the various behavioural aspects being modelled here, so I just remove it for simplicity.

tinuation value function' which is just the exponential discounting value function,

$$\begin{aligned} V(a,z,j) &= \max_{c,aprime} \frac{c^{1-\sigma}}{1-\sigma} + s_j\beta E[V(aprime, zprime, j+1)|z] \\ & \text{if } j < Jr: \ c + aprime = (1+r)a + w\kappa_j z \\ & \text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\ & aprime \geq 0 \\ & log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

we can then define the naive quasi-hyperbolic discounters value function, which we denote $\tilde{V}$ in terms of this,

$$\begin{aligned} \tilde{V}(a,z,j) &= \max_{c,aprime} \frac{c^{1-\sigma}}{1-\sigma} + s_j\beta_0\beta E[V(aprime, zprime, j+1)|z] \\ & \text{if } j < Jr: \ c + aprime = (1+r)a + w\kappa_j z \\ & \text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\ & aprime \geq 0 \\ & log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

There are a few things worth noting. That on the right-hand side we have the 'continuation' next period value function V, while on the left we have the naive quasi-hyperbolic discounters value function $\tilde{V}$. That the discount factor is $\beta_0\beta$, as we are applying the additional discount factor β_0 that captures quasi-hyperbolic discounting.[53]

Now we can think about the sophisticated quasi-hyperbolic discounter. Again we will need to define a 'continuation' value function, but now it is more complex as the sophisticated quasi-hyperbolic discounter knows that their future self will follow the policy function of a sophisticated quasi-hyperbolic discounter, but discount between future periods based on the exponential discount factor (see document linked below for explantion). We will go the reverse order, first defining the value function of the sophisticated quasi-hyperbolic discounter, $\hat{V}$,

$$\begin{aligned} \hat{V}(a,z,j) &= \max_{\hat{c},\hat{a}prime,} \frac{\hat{c}^{1-\sigma}}{1-\sigma} + s_j\beta_0\beta E[\underline{\mathrm{V}}(\hat{a}prime, zprime, j+1)|z] \\ & \text{if } j < Jr: \ \hat{c} + \hat{a}prime = (1+r)a + w\kappa_j z \\ & \text{if } j >= Jr: \ \hat{c} + \hat{a}prime = (1+r)a + pension \\ & \hat{a}prime \geq 0 \\ & log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

notice that I have denoted the policy variables with hats, $\hat{c}$, $\hat{a}prime$. Note that we have the additional discount factor β_0. The continuation value function is now $\underline{\mathrm{V}}$ which is the policies of the sophisticated quasi-hyperbolic

[53] I also apply β_0 to discounting the warm-glow.

discounter but evaluated with the exponential discount factor, and is given by

$$\underline{V}(k,z,j) = \frac{\hat{c}^{1-\sigma}}{1-\sigma} + s_j\beta_0\beta E[\underline{V}(\hat{a}prime, zprime, j+1)|z]$$

where in a slight abuse of notation I am now using $\hat{c}, \hat{a}prime$ to signify the optimal policies (the argmax) from the value function problem of the sophisticated quasi-hyperbolic discounter.

Quasi-hyperbolic preferences are explained in Appendix B.2

7.1.1 Life-Cycle model 36B: Quasi-Hyperbolic Discounting with Endogenous Labor Supply

Adds Quasi-Hyperbolic discounting to Life-Cycle Model 9 (except without warm-glow of bequests).

So for a naive quasi-hyperbolic discounter the continutation value solves,

$$\begin{aligned} V(a,z,j) = & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} + s_j\beta E[V(aprime, zprime, j+1)|z] \\ & \text{if } j < Jr: \ c + aprime = (1+r)a + w\kappa_j zh \\ & \text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\ & aprime \geq 0, \quad 0 \leq h \leq 1 \\ & log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

and we can then define the naive quasi-hyperbolic discounters value function, which we denote $\tilde{V}$ in terms of this,

$$\begin{aligned} \tilde{V}(a,z,j) = & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} + s_j\beta_0\beta E[V(aprime, zprime, j+1)|z] \\ & \text{if } j < Jr: \ c + aprime = (1+r)a + w\kappa_j zh \\ & \text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\ & aprime \geq 0, \quad 0 \leq h \leq 1 \\ & log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

For a sophisticated quasi-hyperbolic discounter the value function is given by,

$$\begin{aligned} \hat{V}(a,z,j) = & \max_{\hat{c},\hat{a}prime,\hat{h}} \frac{\hat{c}^{1-\sigma}}{1-\sigma} - \psi\frac{\hat{h}^{1+\eta}}{1+\eta} + s_j\beta_0\beta E[\underline{V}(\hat{a}prime, zprime, j+1)|z] \\ & \text{if } j < Jr: \ \hat{c} + \hat{a}prime = (1+r)a + w\kappa_j zh \\ & \text{if } j >= Jr: \ \hat{c} + \hat{a}prime = (1+r)a + pension \\ & \hat{a}prime \geq 0 \\ & log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

notice that I have denoted the policy variables with hats, $\hat{c}$, $\hat{a}prime$, $\hat{h}$. The continuation value function is now $\underline{\mathrm{V}}$ which is the policies of the sophisticated quasi-hyperbolic discounter but evaluated with the exponential discount factor, and is given by

$$\underline{\mathrm{V}}(k,z,j) = \frac{\hat{c}^{1-\sigma}}{1-\sigma} - \psi\frac{\hat{h}^{1+\eta}}{1+\eta} + s_j\beta E[\underline{\mathrm{V}}(\hat{a}prime, zprime, j+1)|z]$$

7.2 Life-Cycle model 37: Temptation and Self-Control (Gul-Pesendorfer Preferences)

Under standard preferences, more options is always better. Temptation is the idea that having more options can make you worse off, and self-control captures that you can resist that temptation at a cost. We

We will use Life-Cycle Model 10, and simply change to temptation and self-control (Gul-Pesendorfer) preferences.

$$\begin{aligned} V(a,z,j) = \max_{c,aprime} \frac{c^{1-\sigma}}{1-\sigma} + v(c) - \max \hat{c}v(\hat{c}) \\ +s_j\beta E[V(aprime, zprime, j+1)|z] \\ \text{if } j < Jr: \ c + aprime = (1+r)a + w\kappa_j z \\ \text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\ aprime \geq 0 \\ log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

where $v(c)$ is the 'temptation function'. Notice that we add $v(c)$ based on the same choices as for the value function, and at the same time we subtract the $\max \hat{c}v(\hat{c})$ (note $\hat{c}$, which is entirely independent of c). The interpretation is that the $-\max \hat{c}v(\hat{c})$ is the cost of the temptation coming from the most tempting alternative, and so $\max \hat{c}v(\hat{c}) - v(c)$ is the cost of resisting the temptation.

Implementing this in the codes is just a matter of saying that you are using Gul-Pesendorfer preferences, *vfoptions.exoticpreferences='GulPesendorfer'*, and then defining the temptation function $v(c)$ in the same manner as you would define the return function using *vfoptions.temptationFn*. Note that the return function here remains just the $\frac{c^{1-\sigma}}{1-\sigma}$ (it does not include $v(c)$).

We set the temptation function to be much the same as the return function, namely $v(c) = scaletempt * \frac{c^{1-\sigma_{tempt}}}{1-\sigma_{tempt}}$. You could use any function here, it is just easy to use the same as an example.

For an explanation of how Gul-Pesendorfer preferences conceive of and implement temptation and self-control, see Appendix B.3.

7.2.1 Life-Cycle model 37B: Gul-Pesendorfer Preferences with Endogenous Labor

Adds Gul-Pesendorfer preferences to Life-Cycle Model 9 (except without warm-glow of bequests). Assumes that temptation is in consumption (and not in leisure).

$$
\begin{aligned}
V(a,z,j) \quad = \quad & \max_{hc,aprime} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + v(c) - \max \hat{c} v(\hat{c}) \\
& \qquad + s_j \beta E[V(aprime, zprime, j+1)|z] \\
& \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j z h \\
& \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\
& aprime \geq 0, \quad 0 \leq h \leq 1 \\
& log(zprime) = \rho_z log(z) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon,z})
\end{aligned}
$$

7.3 Life-Cycle model 38: Loss Aversion (Prospect Theory)

Prospect Theory is a way to model loss aversion. With standard preferences people like gains, and dislike losses. Loss aversion is that people dislike losses more than they like gains. Losses and gains are defined relative to a 'reference point', and in our model the reference point is the lag of consumption. There are two keys to setting up loss-aversion: setting up the reference point as a 'residual asset', and using a specific return function so that there is loss-aversion relative to the reference point.

First, we need to add the lag of consumption as an endogenous state, but it is a 'residual asset' in the sense that once we choose our other decisions and next-period endogenous states we have already residually determined next period consumption lag (which is of course just this period consumption). The codes take advantage of setting it up as a residual asset.

Second, for to implementing prospect theory we need a period-utility which incorporates the loss-aversion relative to the reference point. We use the period-utility $U(c, c_{lag})$. This is built around the CES utility function $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$. We define $\Delta = u(c) - u(c_{lag})$, so $\Delta \geq 0$ is a gain, and $\Delta < 0$ is a loss.[54] We then define period-utility as $U(c, c_{lag}) = \theta u(c) + (1-\theta) v(u(c) - u(c_{lag}))$, where $v()$ implements the loss-aversion as $v(\Delta) = (1 - \exp -\mu\Delta)/\mu$ for $\Delta \geq 0$, and $v(\Delta) = -\lambda(1 - \exp(\upsilon/\lambda)\Delta)/\upsilon$ for $\Delta < 0$. The parameter θ is the (inverse of the) importance of the loss aversion, relative to the standard utility. μ controls how quickly the sensitivity of gains decreases at the margin with larger gains (similarly for υ and losses). λ indexes the degree of loss aversion, effectively determining the importance of losses relative to gains.

[54] It is important here that $u(c)$ is strictly monotone in c.

$$
\begin{aligned}
V(a, c_{lag}, z, j) \quad = \quad & \max_{c, aprime} U(c, c_{lag}) + s_j \beta E[V(aprime, c, zprime, j+1)|z] \\
& \text{if } j < Jr: \ c + aprime = (1+r)a + w\kappa_j z \\
& \text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\
& aprime \geq 0 \\
& log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z})
\end{aligned}
$$

note that c is equal to c'_{lag}, hence why it appears in next-period value function.

Implementing this in the codes we need this second endogenous state, c_{lag}, but fortunately it is a 'residual asset'. We therefore set *vfoptions.residualasset=1* and *simoptions.residualasset=1*. The return function and functions to evaluate take as inputs $(d, aprime, a, r, z, ...)$ (omit d if there isn't one). We also need to set up $rprimeFn$, see the codes for how this works, but the generate concept is that it should output the next period value of the residual asset, and that it takes as inputs $(d, aprime, a, z, ...)$ (note, it does not depend on it's own present value).

Note that from the perspective of the toolkit, loss-aversion is really just a specific setup for the return function (specically the utility function). Hence why we do not need to tell the toolkit that we are using prospect theory to implement the loss-aversion. We did need to use a residual asset to implement the reference point (that determines what is a loss/gain), but mathematically we could define the reference point however we like (obviously we want something economically reasonable); residual assets are not specific to prospect theory and can be used for various other purposes.

7.4 Life-Cycle model 39: Ambiguity Aversion

Risk is 'known unknowns', we know all the possible future states and assign a probability to each of them. Ambiguity aims to capture the idea of 'unknown unknowns', we still know all the possible future states, but we cannot assign a specific probability to each of them.

In practice, ambiguity is modelled as 'multiple priors'. Each prior assigns (different) specific probabilities to each possible future state, and we then behave based on the 'worst case' (minimum) across our multiple priors. Notice that ambiguity aversion is solely about the way we form expectations about the future, the way the present is handled is unchanged.

We will introduce ambiguity aversion to life-cycle model 21. The reason for this choice is that this model has a markov shock which is an earnings shock during working age, and then a medical shock during retirement. We will set it up so that the earnings shock is a risk, while the medical shock is an ambiguity. The trick is that risk is just one prior, and we can implement this by simply having all the multiple priors be identical. We set the number of priors for the ambiguity to three (*vfoptions.n_ambiguity*=3), and these priors differ for the medical shock so

that is an ambiguity, but all three priors are identical for the earnings shock so that is a risk (we then show how you can refine this by setting an age-dependent vector for *vfoptions.n_ambiguity* which speeds the computation).

Our household problem is thus,

$$\begin{aligned} V(a,z,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\ & \qquad + s_j \beta \min_{\pi(z)\in\mathcal{P}} E^{\pi(z)}[V(aprime, zprime, j+1)|z] \\ & \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j z h \\ & \text{if } j >= Jr: \; c + aprime = (1+r)a - z + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \\ & zprime = \pi_z(z), \text{ is a two-state markov} \end{aligned}$$

Observe how our expecations are now based on the minimum across the multiple priors $\min_{\pi(z)\in\mathcal{P}} E^{\pi(z)}[V(aprime, zprime, j+1)|z]$, where $\mathcal{P}$ is the set of multiple priors (in practice, different transition matrices for z). $E^{\pi(z)}$ denotes the expectation taken using the prior (transition matrix) $\pi(z)$. Note that during working age ($j < Jr$) z is an employment/unemployment shock, while in retirement z is a medical shock (at all ages it can take two values).

Implementing this in the codes we first need to say that we are using ambiguity aversion by setting *vfoptions.exoticprefererences='AmbiguityAversion'*. Then define the number of multiple priors as *vfoptions.n_ambiguity*. Lastly define the (different) transition matrices for each of the multiple priors in *vfoptions.ambiguity_pi_z* (or *vfoptions.ambiguity_pi_z_J* if they depend on age). Note that you still need to define the 'true' process for z in the standard manner (it is not used for the value function, but is used for simulation), and it is required by the codes that all of the multiple priors use the same grid as the true process.

For an explanation of how Ambiguity Aversion works, see Appendix B.4.

8 Additional Endogenous States

All of our examples thus far have had one endogenous state. We now turn to examples with more than one endogenous state, and also to other kinds of endogenous states. Life-Cycle Models 42-43 demonstrate two other types of endogenous state, which the toolkit calls experienceasset, experienceassetu. We actually already saw two other types of endogenous state, the riskyasset in the portfolio-choice models of Life-Cycle Models 31-35, as well as residualasset in Life-Cycle Model 44. Some of these two endogenous state models are quite large and may throw out-of-memory errors on smaller gpus; many use $vfoptions.divideandconquer = 1$ (see Life-Cycle Model 30) which makes them faster and requires less memory, it requires the the problem is monotone.

A standard endogenous asset in VFI Toolkit has a current endogenous state a, and next period endogenous state $aprime$ can be chosen directly (and as a result, $aprime$ and a both appear in the ReturnFn and the FnsToEvaluate). For other endogenous asset types we again have the current endogenous state a, but next periods endogenous state $aprime$ cannot be chosen directly (and as a result, only a appears in the ReturnFn and the FnsToEvaluate).

We already saw the example of 'riskyasset' in the portfolio-choice problems in Life-Cycle Models 31-35, where $aprime(d,u)$, that is next period endogenous state is a function of decision variable d and i.i.d. between-period shock u.

'experienceasset' models $aprime(d,a)$, that is next period endogenous state is a function of the current endogenous state a and a decision variable d. This can be used for anything that cumuluates based on current decisions, and in Life-Cycle Model 41 we will use it for 'female labor force partipation history'. 'experienceassetu' models $aprime(d,a,u)$, that is next period endogenous state is a function of the current endogenous state a a decision variable d and an i.i.d. between-period shock u. This can be used for anything that cumuluates with uncertainty and based on current decisions, and in Life-Cycle Model 42 we will use it for uncertain human capital. While the main uses of 'experienceasset' and 'experienceassetu' are modelling human capital and uncertain human capital, the can of course be used for anything where $aprime(d,a)$ and $aprime(d,a,u)$ are relevant, respectively.

When using any alternative endogenous state together with a standard endogenous state, the ordering is that the standard endogenous state, call it $a1$, is first and the alternative endogenous state, call it $a2$ is second. So in the ReturnFn and FnsToEvaluate we will have $a1prime, a1, a2$ (recall alternative states mean we cannot choose $a2prime$ directly); with decision variables before these and exogenous states after these. If the model has multiple decision variables, the 'last' decision variables will be understood as those that determine the evolution of the alternative endogenous state.

Summarizing, we have

- riskyasset: $aprime(d,u)$
- experienceasset: $aprime(d,a)$
- experienceassetu: $aprime(d,a,u)$

- residualasset: $a2prime(d, a1prime, a1)$

For two standard endogenous states, the codes restrict all choices onto the grid, and so you need lots of points (as in the earlier one endogenous state models; you can use divide-and-conquer for the first of the two assets, and this is done in the example). For experienceasset and experienceassetu, while the decision variable is chosen on the grid, the value for *aprime* is evaluated using linear interpolation, and so you can use way less points on experienceasset and experienceassetu than you would on a standard endogenous state.

8.1 Life-Cycle model 42: Female Labor Force Participation History (experienceasset)

The household will have two endogenous states, the first is a standard endogenous state a, and the second is female-labor-force-participation-history h which is an 'experienceasset'. Each period the household will decide whether the female works or not (a binary decision variable, p), and the female-labor-force-participation-history will evolve based on this decision, $hprime(p, h)$.

The household represents a couple. For simplicity the male always works (for earnings y_m). The household can save assets a, and the female gets earnings $y_f = phz$ where p is the labor-force-participation decision (1 is working, 0 if not working), h is female-labor-force-participation-history, and z is a markov process on female productivity (per unit of human capital).

Empirically, perhaps the most important difference between male and female labor force participation comes around the birth of children. The parameter $childcarecosts_j$ is zero at most ages, but we make in non-zero during ages 27-31, representing having infants/young children during these ages. Females decision on whether to work will depend on their potential earnings, the disutility of working, φp, and childcare costs (if relevant given age).

The rest of the household problem is as close as possible to earlier examples, and is,

$$\begin{aligned}
V(a,h,z,j) \quad = \quad & \max_{c,aprime,p} \frac{c^{1-\sigma}}{1-\sigma} - \varphi p + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\
& \qquad + s_j \beta E[V(aprime, hprime, zprime, j+1)|z] \\
& \text{if } j < Jr: \; c + aprime = (1+r)a + y_{m,j} + y_f - childcarecosts_j p \\
& \qquad y_f = phz \\
& \qquad hprime = exp(log(h) + h_{accum} * p - \delta_h * (1-p)) \\
& \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\
& p \in \{0,1\}, aprime \geq 0 \\
& log(zprime) = \rho_z log(z) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon,z})
\end{aligned}$$

note that the female-labor-force-participation history (human capital, modelled as an experienceasset) evolves

according to $hprime = exp(log(h) + h_{accum} * p - \delta_h * (1-p))$, which is saying that if the female works then h increases by h_{accum} and if she does not work then h decreases by δ_h. This is $aprime(d,a)$ for the experienceasset, and we will set it up in *vfoptions.aprimeFn*.

This kind of model is used to understand decision around female labor-force participation. Here children were exogenous, but you can make decisions around having children endogenous as a semi-exogenous state as seen in Life-Cycle Model 29. These models will also often include a time cost of young children (not just the monetary cost of childcare), which is omitted here to keep the model as simple as possible.

More generally, experienceasset will be useful for any kind of model where you want next period endogenous state to be a function of this period endogenous state and a decision variable, so $aprime(d,a)$. Note that it is important when setting up *n_a* that the endogenous state is the second asset, and for the grids you create *a_grid* as a stacked column vector of the grids for each of the two assets. And that if you have multiple decision variables, when setting up *n_d* the decision that matters for the experienceasset must be the last one, and for the grids you create *d_grid* as a stacked column vector of the grids for each decision variable. You can also use an experienceasset in a one endogenous state model (as the only endogenous state).

8.2 Life-Cycle model 43: Uncertain Human Capital (experienceassetu)

The household will have two endogenous states, the first is a standard endogenous state a, and the second is human capital h which is accumulated with uncertainty, and hence an 'experienceassetu'. Each period the household will decide what fraction of time to spend studying (a decision variable, s) and the human capital will evolve based on this decision $hprime(d,h,u)$ with uncertainty due to a between-period i.i.d shock, u. We also have two decision variables, l, the labor supply, and s the time spent studying. When using experienceassetu with multiple decision variables, the last decision variable will be the one that influences the experienceassetu.

The household problem is,

$$\begin{aligned} V(a,h,z,j) \quad = \quad & \max_{c,aprime,s,l} \frac{c^{1-\sigma}}{1-\sigma} + \varphi \frac{leisure^{1-\eta}}{1-\eta} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\ & \qquad + s_j \beta E[V(aprime, hprime, zprime, j+1)|z] \\ & \text{if } j < Jr: \ c + aprime = (1+r)a + whlz \\ & \qquad hprime = u(ability(sh)^{\alpha_h} + (1-\delta_h)h) \\ & \text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\ & leisure + s + l = 1, aprime \geq 0 \\ & log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \\ & log(u) \sim N(0, \sigma^2_{\epsilon,u}) \end{aligned}$$

where l is labor supply and s is time spent studying.

The human capital production function is $hprime = u(ability(sh)^{\alpha_h} + (1-\delta_h)h)$, and it is this that we set up as *vfoptions.aprimeFn* for $aprime(d,a,u)$.

More generally, experienceassetu will be useful for any kind of model where you want next period endogenous state to be a function of this period endogenous state, a decision variable, and a between-period iid shock, so $aprime(d,a,u)$. Note that it is important when setting up n_a that the endogenous state is the second asset, and for the grids you create a_grid as a stacked column vector of the grids for each of the two assets. And that if you have multiple decision variables, when setting up n_d the decision that matters for the experienceassetu must be the last one, and for the grids you create d_grid as a stacked column vector of the grids for each decision variable. You can also use an experienceassetu in a one endogenous state model (as the only endogenous state).

9 Estimating Life-Cycle Models

Now that we understand how to set-up and solve Life-Cycle Models, an obvious question is how to choose the parameters. The most common approach is calibration, which is necessarily a somewhat ad-hoc process. The alternative is to use formal statistical estimators. The two most common statistical estimators used for life-cycle models are Generalized Method of Moments (GMM) and Maximum Likelihood Estimation (MLE); other options are Indirect Inference and Bayesian Likelihood Estimation. We now show how to perform GMM estimation of life-cycle models.[55]

The core idea of GMM estimation of Life-Cycle models that we will choose the model parameters, θ, to get moments of the model $M^m(\theta)$ as close as possible to the moments of the data M^d, based on minimizing the sum-of-squares difference and using a weighting matrix W. That is, we estimate θ as,

$$\theta^* = \arg\max_{\theta} (M^d - M^m(\theta))' W (M^d - M^m(\theta))$$

and under certain assumptions the estimate is asymptotically normal

$$\sqrt{(N)}(\hat{\theta} - \theta_0) \to N(0, \Sigma)$$

where $\Sigma = (J'WJ)^{-1} J'W'\Omega WJ(J'WJ)^{-1}$ is the covariance matrix of the estimated parameter vector (so the standard deviation of the parameter estimates is the square root of the diagonal elements of Σ). J is the Jacobian matrix of the derivatives of the model moments with respect to the estimated parameter vector, and Ω is the covariance matrix of the data moments.

GMM estimation in VFI Toolkit is implemented by the command *EstimateLifeCycleModel_MethodOfMoments* (and *EstimateLifeCycleModel_PType_MethodOfMoments*). Life-Cycle Models 45-50 provide examples covering various aspects of GMM estimation. Before estimating a Life-Cycle model you will need to work with real-world data to get estimates of the target moments, M^d, and their variance-covariance matrix Ω; Life-Cycle Model 47 provides an example of how to do this in Matlab, but some people will prefer to do this in other applications like Stata or R.

The first four examples, Life-Cycle Models 45, 46, 47 and 48, are all based on GMM estimating Life-Cycle Model 9 (endogenous labour, and an exogenous markov shock), and the fifth and sixth examples shows how to GMM estimate the extension of this model to permanent types. Life-Cycle Model 45 just does a baseline GMM estimation, to illustrate the core concepts and commands. Life-Cycle Model 46 shows how we can include parameters that determine exogenous shocks and the initial agent distribution among the parameters to be estimated, and also show how to restrict parameters to be, e.g., positive, or valued between 0 and 1. Life-Cycle Model 47 is about how

[55] VFI Toolkit cannot presently do any other type of statistical estimator. That calibration is somewhat ad-hoc does not mean I dislike/avoid calibration. Calibration is a good option. It is simply an observation that there are no agreed formal rules on how calibration should be performed.

to use the data to create the data moments and the covariance matrix of the data moments, and also about how we choose the weighting matrix used in GMM, describing both a simple/robust way to avoid scale dependence, and also how to implement efficient GMM. Life-Cycle Model 48 is just a repeat of 46, but this time showing various options and going through the different outputs to explain what they are. Life-Cycle Model 49 shows how to handle permanent types by GMM estimating a minor extension of Life-Cycle Model 9 to include a fixed-effect in potential hourly-earnings, this example uses permanent types to estimate unobserved heterogeneity. Life-Cycle Model 50 is also permanent types with GMM, but this time we will have targets that are conditional on permanent type (some model parameters differ by type, some are common across type).

To be able to GMM estimate any model we must provide the target data moments, M^d, the weighting matrix W, and the covariance matrix of the data moments Ω. Life-Cycle Model 47 shows an example of how to get these three from the data. The rest of the examples skip the data work and just include values for these three, so as to focus on how the estimation is done in VFI Toolkit. This is done as there are plenty of explanations out there about how to do the data work, and this document is about how to do the computation and estimation of the model.

Appendix C gives a brief formal explanation of GMM and how it is applied to Life-Cycle Models.

The GMM Estimation of Life-Cycle Models is based around the same value function and agent distribution commands as all the examples in this Intro to Life-Cycle Models. This means that every model we have seen —from Epstein-Zin preferences, to semi-exogenous shocks, to 'riskyasset', to 'experienceasset'— can be GMM estimated in exactly the same way.[56]

9.1 Life-Cycle model 45: GMM Estimation, the basics

We will GMM estimate Life-Cycle model 9. We make one change, namely to the initial age $j = 1$ agent distribution; previously it was singular, which is not good for doing statistical estimation. We instead use a joint-normal distribution on (a, z) as the initial age $j = 1$ distribution. The model to be estimated is the household value function,

$$\begin{aligned} V(a,z,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\ & \qquad + s_j \beta E[V(aprime, zprime, j+1)|z] \\ & \text{if } j < Jr: \ c + aprime = (1+r)a + w\kappa_j zh \\ & \text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \\ & log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \end{aligned}$$

[56] VFI Toolkit can GMM estimate any model that uses *ValueFnIter_Case1_FHorz()* with *EstimateLifeCycleModel_MethodOfMoments*, and any model that uses *ValueFnIter_Case1_FHorz_PType()* with *EstimateLifeCycleModel_PType_MethodOfMoments'*.

with initial agent distribution joint-normally distributed over assets and exogenous shocks, $\lambda_1(a,z) = N([\mu_{1,a}; \mu_{1,z}], \Sigma)$, with variance-covariance matrix Σ.

GMM estimation requires us to decide which parameters to estimate —we will estimate the preference parameters σ, η and ψ— and which moments to target —we will target the age-conditional mean earnings (for working ages). We also need to pick a weighting matrix —we will use the identity matrix— and we need to include the variance-covariance matrix of the data moments so that we can compute standard deviations and confidence intervals for the estimated parameters. We discuss how to actually estimate the target data moments, their covariance matrix, and how to choose a weighting matrix in Life-Cycle Model 47; for now they are just given in the codes.

9.2 Life-Cycle model 46: GMM Estimation, parameter restrictions, estimating shocks and initial dist

We repeat the GMM estimation exercise from Life-Cycle Model 45, but with some changes. First, we show how by setting up exogenous shock processes and initial agent distributions as parametrized functions we can estimate these as part of the GMM estimation. Second, we show how a parameter to be estimated can be restricted, e.g., to be positive valued.

There are three types of parameter restrictions: (i) we can restrict a parameter to be positive, (ii) we can restrict a parameter to be between 0 and 1, (iii) we can restrict a parameter to be between any to constants, A and B.[57] All of these are trivial to implement using *estimoptions*. To restrict parameters to be positive, we set *estimoptions.constrainpositive* to contain the names of the parameters we wish to restrict to be positive. To restrict parameters to be between 0 and 1, we set *estimoptions.contrain0to1* to contain the names of the parameters we wish to restrict to be positive. To restrict parameters to be between A and B, we set *estimoptions.contrainAtoB* to contain the names of the parameters we wish to restrict to be between A and B, and then use *estimoptions.contrainAtoBlimits* to set the values of A and B.

To estimate the exogenous shock process we set up *vfoptions.ExogShockFn* (and set *simoptions.ExogShockFn=vfoptions.ExogShockFn*) to take in parameter values, and output [z_grid,pi_z].[58]

To estimate the initial agent distribution we set up *jequaloneDistFn* to take inputs *(a_grid,z_grid,n_a,n_z,...)*, where the ... are any parameters. We can then pass this *jequaloneDistFn* to anywhere we would normally pass *jequaloneDist*.

We set up the initial distribution as being joint log-normal on assets and the markov shock. We include the

[57] Say the parameter is θ. VFI Toolkit imposes restrictions by reformulating the problem as an unconstrained problem. So if we restrict parameter θ to be positive, then internally we instead work with estimating $\mu = log(\theta)$, which can take any value, and the model is passed $exp(\mu)$. If we restrict a parameter θ to be zero to one, then internally we instead work with estimatiing $\mu = log(\theta/(1-\theta))$, which can take any value, and the model is passed $1/(1+exp(-\mu))$. If we restrict a parameter θ to be A to B, we do the same as for 0 to 1, but with additional step before of $(\theta - A)/(B - A)$, and after of $A + \theta * (B - A)$ to convert between A-to-B and 0-to-1, and back again.

[58] This can handle age-dependent shocks. If you use and i.i.d. 'e' variable, you can similarly set *vfoptions.EiidShockFn* (and set *simoptions.EiidShockFn=vfoptions.EiidShockFn*).

parameters of the covariance matrix as part of the parameters to be estimated. Because covariance matrices have to be positive semi-definite doing this in a naive way would generate lots of errors. So we follow a method to parameterize covariance matrices that ensures they are always positive semi-definite. This is included as covariance matrices are often something people want to estimate in practice so it is good to see how to do this (the covariance matrix is parametrized with three parameters, but only two will be estimated because the third is not identified by the target moments used in this example).

Our estimation is of the same three preference parameters as previously —σ, η and ψ— as well as the two parameters of the Markov shocks —ρ_z and $\sigma_{\epsilon,z}$— and two parameters for the covariance matrix of initial distribution of agents (see code for explanation of how we parameterize the covariance matrix of the initial distribution).

To make it more likely that we can identify so many parameters, we still target the age-conditional mean earnings, but add further targets for the age-conditional labor supply. We continue to use the identity matrix as a weighting matrix, and the covariance matrix of the data moments is also adjusted. We discuss how to actually estimate the target data moments, their covariance matrix, and how to choose a weighting matrix in Life-Cycle Model 47; for now they are just given in the codes.

9.3 Life-Cycle model 47: GMM Estimation, using data and how to choose Weighting Matrix

We now reestimate the same model as in Life-Cycle Model 45, we again target age-conditional mean earnings but the parameters to be estimated is κ_j, the deterministic age-dependent component of earnings. The big difference will be that this time we use real-world data. We show how to use panel data to get the data moments and the covariance matrix of the data moments. Specifically, we use the PSID panel data to estimate the age-conditional mean earnings, and the covariance matrix of these. We also consider three different choice for the weighting matrix and discuss the pros and cons of these in terms of robustness, statistical efficiency, and scale-independence.

Once we have a Life-Cycle Model that we want to estimate, and have decided what data moments we want to target (by getting the model moments to match them) there are two main things we need to calculate from the data. The first is the data moments themselves. The second is the covariance matrix of the data moments.

For some simple examples of how to perform GMM estimation without the complication of the Life-Cycle Model, so based on OLS and AR(1), see

github.com/robertdkirkby/GMMSeparableMoments/blob/main/GMMexamples.pdf

Because there is a lot less going on in these examples it is much easier to seen how to calculate the data moments, and then how to calculate the covariance matrix of the data moments.

We have observable data $\{x_i\}_{i=1}^N$; so N data observations. Say we have m target moments, so $M^d = \frac{1}{N}\sum_{i=1}^n f(x_i)$, where $f(x_i)$ is a vector of size $m - by - 1$ (and is a function of a single data observations x_i).

For our present application, our target moments are the age-conditional mean earnings, and any observation can be thought of as earnings and age, so $x_i = (y_i, j_i)$, where y_i is the earnings and j_i is the age. So for us $f(x_i) = f(y_i, j_i) = [y_i \mathbb{1}_{(j_i==j)}]_{j=1}^{j=J}$, where $[]_{j=1}^{j=J}$ indicates that it is a $J-by-1$ vector, with each element corresponding to a specific age (our application has $m = J$). So we have $M^d = \frac{1}{N}\sum_{i=1}^{n}[y_i \mathbb{1}_{(j_i==j)}]_{j=1}^{j=J}$, and which is a $(m =)$ J-valued vector. That is the data moments, how about the covariance of the data moments, Ω. We can estimate this as $\hat{\Omega} = \frac{1}{N}\sum_{i=1}^{n} f(x_i)f(x_i)'$, which is a $m-by-m$ matrix ($f(x_i)$ is $m-by-1$ so $f(x_i)f(x_i)'$ is $m-by-m$); we already saw what $f(x_i)$ is in our application.[59]

In the code, we start by downloading and importing PSID data.[60] We then impose some sample restrictions. In the data, there are age-cohort-time effects and it is only possible to control for any two of these (you can see this as if I tell you any two of a person's current age, the current year (time), and the person's year of birth (cohort), then you can tell me the third). We therefore first estimate the mean age-conditional earnings based on cohort fixed-effects, and calculate the corresponding covariance matrix. This is what we then use in our GMM estimation of the Life-Cycle Model 48. We also calculate the mean age-conditional earnings based on time fixed-effects, just to show how they differ (they are not used in estimation of the model).

We now have the data moments and the covariance matrix of the data moments. Next, we need to sort out what parameter we want to estimate.

The parameter we want to estimate is κ_j, the deterministic age-dependent component of earnings (earnings is $w\kappa_j zh$). Note that as there are 45 working-age periods we are estimating 45 values for κ_j. The code contains a variable 'parametrizeKappaj' which is equal to one by default: when it equals one we will parametrize (the log of) κ_j as a fifth-order polynomial and estimate the coefficients of this polynomial, when it equals zero we just estimate the vector κ_j itself. Parametrizing (log) κ_j as a polynomial is common in practice, but the codes show both alternatives. To parameterize a parameter we use 'ParametrizeParamsFn', which is simply a function that takes the parameter structure as an input and returns the parameter structure as an output. Note that we want κ_j to be positive, hence it is $log(\kappa_j)$ which we parametrize with a fifth-order polynomial.[61]

We now have the data moments and the covariance matrix of the data moments. And we have the parameters to be estimated set up. So we are ready to GMM estimate Life-Cycle Model 47. We just need to choose a weighting matrix.

Choosing the Weighting Matrix As long as we use a positive-semidefinite weighting matrix W, GMM estimation will be consistent and asymptotically normal. Until now we just used $W = \mathbb{I}$ for simplicity, but this is a poor choice.

[59] This is all based on GMM theory for i.i.d data. Hence the resulting $\hat{\Omega}$ ends up with zeros on the off-diagonal because the rows/columns are for different ages. Really, this is incorrect because the data is not i.i.d., it is panel data, but it is standard practice for estimating life-cycle models. We are not aware of theory for how to calculate $\hat{\Omega}$ for panel data (with cross-section going to infinity, while time-horizon is fixed); there is GMM theory for time series, and then $\hat{\Omega}$ should account for autocorrelation (e.g., use Newey-West estimator of covariance matrix).

[60] Using ImportPSIDdata(), which is part of VFI Toolkit.

[61] The fifth-order polynomial is in age/100, rather than in age, the /100 is just that otherwise small changes in the co-efficients lead $exp(age^5)$, which is essentially one of the terms of the polynomial, to blow up to infinity.

The size of the confidence intervals on our estimated parameters depends on W. As Appendix C explains, the optimal weighting matrix —the weighting matrix that minimizes the size of the variance-covariance matrix of the estimated parameters— is given by $W = \Omega^{-1}$. Choosing $W = \Omega^{-1}$ is called efficient GMM. We show how to implement this.

In small samples the optimal weighting matrix $W = \Omega^{-1}$ can sometimes lead to a biased estimator. And so we might want something simplier, in which case $W = diag(\Omega)^{-1}$ (the inverses of the diagonal elements of Ω, with zeros on the off-diagonals) is a popular choice as it is scale independent. Notice that if we calculate the difference between data mean income and model mean income, square it, and multiply by one (with the identity matrix as our weighting matrix) then the resulting answer is very different if we use dollars or thousands of dollars; this is scale dependence, if we use different units for different moments then the scales of these units changes our estimated parameter vector. By using $W = diag(\Omega)^{-1}$ we eliminate scale dependence; notice that the weight for each moment is going to divide by the square of the units, thus eliminating the units.

In the codes we first estimate $\hat{\Omega}$, and we then do the estimation three times, once using $W = \mathbb{I}$ once using $W = diag(\hat{\Omega})^{-1}$, and once performing efficient GMM using $W = \hat{\Omega}^{-1}$.

Note that regardless of which weighting matrix we use, we need to estimate Ω in any case as it is needed as an input to the estimation command because it appears in the formula for calculating Σ, the covariance matrix of the estimated parameters.

9.4 Life-Cycle model 48: GMM Estimation, various extras

We largely just reperform the estimation from Life-Cycle model 45, but this time looking at various *estimoptions* and some of the other outputs.

First, instead of targeting the same moments, for half of them we target the log of the moment. To do this we set *estimoptions.logmoments* to be a column vector zeros of the length of the vector of moments to be estimated, but with ones in the elements that correspond to the moments we wish to restrict to be positive.

Second, rather than targeting (log) age-conditional mean earnings we instead target just the even ages, this just involves putting NaN in the target moments for any that we want to ignore (in our case all the odd ages). This particular example is rather stupid, and is just to illustrate how you can use NaN to omit any moments you don't want.

Third, the default is reporting 90-percent confidence intervals for the estimated parameters. We can change this using, e.g., *estimoptions.confidenceintervals=95* to get the 95-percent confidence intervals. If we are interested in standard deviations of the estimated parameters, these can be found in *estsummary.EstimParamsStdDev*.

Fourth, we look at whether the model parameters are locally-identified. A standard approach is to look at the rank of J, and is reported in *estsummary.localidentification*.

Fifth, we are interested in which moments are determining which parameters. *estsummary.sensitivitymatrix* reports a sensitivity matrix; rows index the parameters, column index the moments, and a large number (relative

to parameter value) indicates that this parameter is sensitive to this moment.

Sixth, we might be interested in how sensitive our estimated parameters are to any pre-calibrated parameters. Setting *estimoptions.CalibParamsNames* to, e.g., *estimoptions.CalibParamsNames*={*'w'*} , means there will be a *estsummary.sensitivitytocalibrationmatrix* which reports the sensitivity of estimated parameters (rows) to the pre-calibrated parameters (columns).

Seventh, we might be interested in the derivatives of various model moments at given parameter values and the command 'EstimateLifeCycleModel_MomentDerivativess' calculates these. For example, these derivatives are important to identification, and for the width of the confidence intervals. Massively oversimplifying, the width of the confidence intervals of the estimated parameter will be the variance of the target moment divided by the derivative of the moment with respect to the estimated parameter. So if we want to estimate a parameter then, conditional on the variance of the moments, we want to target moments where these derivatives are large. In principle we are interested in these derivatives evaluated at the estimated parameter values, but since we won't know these before estimation we just evaluate them as some initial guesses for the parameters. Note, this command is essentially calculating J, but for lot's of moments, and at some initial parameter values rather than the estimated parameters. These derivatives can help guide decisions on what to target, in particular, moments with big derivatives for a given parameter are going to be better at identifing and giving small confidence intervals, conditional on the variances of the moments.

Finally, just a few *estimoptions* we don't use. *estimoptions.verbose* is set to one by default, if you set it to zero you will get less feedback on the estimation. *estimoptions.skipestimation* can be set to one, this keeps the same parameter vector as is normally input as an initial guess, but recomputes all the outputs (other that the parameter vector); can be used to add sensitivity analysis for calibrated parameters later on, or just to recompute standard errors using a more accurate version of the model (using larger grids). *estimoptions.fminalgo* by default equals 4, which uses CMA-ES algorithm, which is very robust but not so fast; setting to 1 uses fminsearch() but in my experience this often failed to converge (GMM estimation can be a difficult optimization problem). *estimoptions.toleranceparams* and *estimoptions.toleranceobjective* can be used to set the tolerance used to decide when a solution has been found.

Note that Appendix C has formulae and explanations of many of these concepts: The formula for the covariance matrix of the estimated parameter vector. How to calculate standard deviations of the estimated parameters from the covariance matrix of the estimated parameter vector. How to calculate confidence intervals from the standard deviations. How to evaluate local identification. How to calculate local sensitivity of the estimated parameters to the target moments. How to calculate local sensitivity of the estimated parameters to the pre-calibrated parameters.

9.5 Life-Cycle model 49: GMM Estimation, permanent types as unobserved heterogeneity

We modify Life-Cycle model 9 to have permanent types, specifically a fixed-effect in hourly-earnings. This requires a minorly different GMM Estimation command to be used (as is standard in VFI Toolkit when using permanent types).

When doing GMM Estimation of Life-Cycle Models with permanent types, you can choose to just use moments that are across all permanent types, or choose to use moments that are conditional on permanent type, or a mixture of the two. In this model we will use a moments that do not depend on permanent type (they are an average across permanent types) and we will see how to use permanent types for unobserved heterogeneity. In Life-Cycle Model 50 we will show how to use targets that are conditional on permanent type.

Since the model is new we now describe it in full.

$$
\begin{aligned}
V(a,z,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\
& \qquad + s_j \beta E[V(aprime, zprime, j+1)|z] \\
& \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j \alpha_i z h \\
& \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\
& 0 \leq h \leq 1, aprime \geq 0 \\
& log(zprime) = \rho_z log(z) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon,z}) \\
& log(\alpha_i) \sim N(\bar{\alpha}, \sigma^2_{\alpha})
\end{aligned}
$$

with initial agent distribution joint-normally distributed over assets and exogenous shocks, $\lambda_1(a,z) = N([\mu_{1,a}; \mu_{1,z}], \Sigma)$, with variance-covariance matrix Σ.

The change from Life-Cycle Model 9 (as used in Life-Cycle Model 45) is the addition of α_i. We will use $N\_i = 5$ permanent types. Normally what we would do is discretize (using Farmer-Toda) method the normal distribution on (log) α_i into five points with associated probabilities, and then store these in the parameter structure. We will parametrize the distribution, as $\bar{\alpha}$ and σ_α, and then estimate these two parametrize.

Our target moments will be the mean of age-conditional earnings, and the variance of age-conditional earnings (roughly, the mean should help for $\bar{\alpha}$ and the variance for σ_α).

To be able to estimate unobserved heterogeneity we need to be estimating the distribution of the permanent types as part of the GMM estimation. Note that the distribution of permanent types is typically kept in the parameters structure, and we are parametrizing it by $\bar{\alpha}$ and σ_α which are also being kept in the parameters structure. To do this we use 'ParametrizeParamsFn', which is simply a function that takes the parameter structure as an input and returns the parameter structure as an output; so inside the function we will use $\bar{\alpha}$ and σ_α in the Farmer-Toda method to discretize α_i into values (a grid) and probabilities, both of which we store in the parameter

structure for output. Essentially, 'ParametrizeParamsFn' is used to update our permanent types based on the current values of the parametrization (of $\bar{\alpha}$ and σ_α).

Note, an alternative approach (but more complicated) is to include the permanent type masses/probabilities as part of the initial agent distribution, in which case we would have to parametrize and estimate the initial agent distribution like we saw in Life-Cycle Model 46. You would need to do this if, e.g., the permanent types were correlated with inital asset holdings.

Two comments on the estimation itself. First, normally you would not be able to tell the mean of α_i appart from adding a constant to κ_j, so you would probably just be normalizing α_i to be mean zero rather than estimating it. Second, the results show that while $\bar{\alpha}$ is fairly well estimated, σ_α is not. Notice that while σ_α does matter for the variance of earnings, z plays a much larger role in the variance of earnings (given the parametrization of our model) and so the estimation struggles to pin down σ_α. This is deliberately left here to remind you about the importance of thinking about parameter identification.

9.6 Life-Cycle model 50: GMM Estimation, permanent types 2

We will have two permanent types, call them 'funtimes' and 'worktimes'. This example shows two things. First, how to have estimation parameters, some of which are the same across permanent types and some of which are different across permanent types (we kind of already saw this). Second, how to have estimation target moments some of which are the same across permanent types (we saw this already) and some of which are specific to certain permanent types.

Our model is identical to that used in Life-Cycle Model 45, except that we allow the parameter ψ (the weight on leisure in the utility function) to differ across permanent types, specifically it will be larger for 'funtimes' and smaller for 'worktimes'.

We then target the age-conditional mean of earnings (across all permanent types) as well as the age-conditional mean of earnings for 'funtimes' agents and the age-conditional mean of earnings for 'worktimes' agents.

10 What Next?

If you want to learn to solve more complex life-cycle models you will need to learn how to code them yourself, and the numerical methods used to do so. There are a lot of materials online,[62] and a fast method like endogenous grid points for solving the value function problem is probably a good place to start. If you just want to use one asset life-cycle models then the VFI Toolkit is good enough for almost everything, with sole exception of 'simulated likelihood estimation' of the models (typically from panel data).[63]

If you are interested in building Macroeconomic models then OLG models are the next obvious step. In life-cycle models we just have individual households. An OLG model combines the life-cycle model of a household with general equilibrium to create a model of the entire economy. Take a look at: An Introduction to OLG models.

If you have any questions about the material, or spot a typo in the codes, or would just like to ask a clarifying question, etc., please use the forum: discourse.vfitoolkit.com

[62] I rate the material of Fatih Guvenen. Check https://github.com/KennethJudd/CompEcon2020 for the underlying theory on computation, there are no end of more applied resources and where you should look depends heavily on what kind of problem you want to solve.

[63] Simulated likelihood estimation requires solving the model a lot of times, say 1 million times. If you take the run time for the codes here and multiply them by 1 million you will see how long that would take.

References

Isaiah Andrews, Matthew Gentzkow, and Jesse Shapiro. Measuring the sensitivity of parameter estimates to estimation moments. *Quarterly Journalk of Economics*, 132(4), 2017. doi: https://doi.org/10.1093/qje/qjx023.

Timo Boppart and Per Krusell. Labor supply in the past, present, and future: A balanced-growth perspective. *Journal of Political Economy*, 128(1), 2020. doi: https://doi.org/10.1086/704071.

Christopher Carroll. Theoretical foundations of buffer stock saving. *Quantitative Economics*, Forthcoming:AAA, 2021.

Christopher Carroll, Edmund Crawley, Jiri Slacalek, Kiichi Tokuoka, and Matthew White. Sticky expectations and consumption dynamics. *American Economic Journal: Macroeconomics*, 12(3):40–76, 2020.

Simone Civales, Luis Diez-Catalan, and Fatih Fazilet. Discretizing a process with non-zero skewness and high kurtosis. *Unpublished manuscript*, 1:1, 2017.

Juan Carlos Cordoba and Marla Ripoll. Risk aversion and the value of life. *Review of Economic Studies*, 84, 2017. doi: https://doi.org/10.1093/restud/rdw053.

Larry G. Epstein and Stanley E. Zin. Substitution, risk aversion, and the temporal behavior of consumption and asset returns: A theoretical framework. *Econometrica*, 57(4):937–969, 1989.

Leland E. Farmer and Alexis Akira Toda. Discretizing nonlinear, non-gaussian markov processes with exact conditional moments. *Quantitative Economics*, 8(2):651–683, 2017.

Giulio Fella, Giovanni Gallipoli, and Jutong Pan. Markov-chain approximations for life-cycle models. *Review of Economic Dynamics*, 34:183–201, 2019.

Itzhak Gilboa and David Schmeidler. Maxmin expected utility with non-unique prior. *Journal of Mathematical Economics*, 18(2), 1989. doi: https://doi.org/10.1016/0304-4068(89)90018-9.

Gene M. Grossman, Elhanan Helpman, Ezra Oberfield, and Thomas Sampson. Balanced growth despite uzawa. *American Economic Review*, 107:4, 2017.

Fatih Guvenen, Fatih Karahan, Serdar Ozkan, and Jae Song. What do data on millions of u.s. workers say about lifecycle labor income risk? *Econometrica*, 89(5):2303–2339, 2021. doi: https://doi.org/10.3982/ECTA14603.

Cosmin Ilut and Martin Schneider. Modeling uncertainty as ambiguity: a review. In Topa Bachmann and Klaauw, editors, *Handbook of Economic Expectations*, chapter Part 4, Chapter 24. Elsevier, 2023 edition, 2023.

Ayse Imrohoroglu. Cost of business cycles with indivisibilities and liquidity constraints. *Journal of Political Economy*, 97(6):1368–1383, 1989.

Thomas Jorgensen. Sensitivity to calibrated parameters. *Review of Economics and Statistics*, 105(2), 2023. doi: https://doi.org/10.1162/rest_a_01054.

Greg Kaplan. Inequality and the life-cycle. *Quantitative Economics*, 3(3), 2012. doi: https://doi.org/10.3982/QE200.

Fatih Karahan and Serdar Ozkan. On the persistence of income shocks over the life cycle: Evidence, theory, and implications. *Review of Economic Dynamics*, 16(3):452–476, 2013. doi: https://doi.org/10.1016/j.red.2012.08.001.

Karen Kopecky and Richard Suen. Finite state markov-chain approximations to highly persistent processes. *Review of Economic Dynamics*, 13(3):701–714, 2010.

Holger Kraft, Claus Munk, and Farina Weiss. Bequest motives in consumption-portfolio decisions with recursive utility. *Journal of Banking and Finance*, 138, 2022. doi: https://doi.org/10.1016/j.jbankfin.2022.106428.

David Laibson. Golden eggs and hyperbolic discounting. *Quarterly Journal of Economics*, 112(2):443–478, 1997. doi: https://doi.org/10.1162/003355397555253.

Bartosz Maćkowiak and Mirko Wiederholt. Business cycle dynamics under rational inattention. *Review of Economic Studies*, 82(4):1502–1532, 2015. doi: https://doi.org/10.1093/restud/rdv027.

Salvador Ortigueira and Nawid Siassi. How important is intra-household risk sharing for savings and labor supply? *Journal of Monetary Economics*, 2013. doi: https://doi.org/10.1016/j.jmoneco.2013.05.007.

Jeno Pál and John Stachurski. Fitted value function iteration with probability one contractions. *Journal of Economic Dynamics and Control*, 37:251–264, 2013.

Josep Pijoan-Mas. Precautionary savings or working longer hours? *Review of Economic Dynamics*, 9(2):326–352, 2006.

G. Rouwenhorst. Asset pricing implications of equilibrium business cycle models. In Cooley T., editor, *Frontiers of Business Cycle Research*, chapter 10. Princeton University Press, 1995.

Eric Swanson. Risk aversion and the labor margin in dynamic equilibrium models. *American Economic Review*, 102(4):1663–91, 2012.

Eric Swanson. Risk aversion, risk premia, and the labor margin with generalized recursive preferences. *Review of Economic Dynamics*, 29:290–321, 2018.

Kenichiro Tanaka and Alexis Akira Toda. Discrete approximations of continuous distributions by maximum entropy. *Economics Letters*, 118(3):445–450, 2013.

George Tauchen. Finite state markov-chain approximations to univariate and vector autoregressions. *Economics Letters*, 20:177–181, 1986.

George Tauchen and Robert Hussey. Quadrature-based methods for obtaining approximate solutions to nonlinear asset pricing models. *Econometrica*, 59(2):371–396, 1991.

A Alternative exogenous shock processes

These are a variety of life-cycle models that lightly modify Life-Cycle Models 9 and 11 to illustrate a variety of alternative kinds of exogenous shock processes that you could use. I use the works 'exogenous shock' and 'exogenous state' rather interchangeably in this Appendix.

VFI Toolkit understands three types of exogenous state: semi-exogenous markov, exogenous markov, and exogenous i.i.d., which are denoted $semiz$, z and e, respectively. VFI Toolkit always uses the ordering that the shocks come immediately after the endogenous states, and that they follow $semiz$, z, e. You can have up to four of each of $semiz$, z and e, in which case all the shocks of the same type enter in order, so for example if we had a model with one semi-exogenous shock, zero exogenous markovs and three exogenous i.i.d., then the inputs for the ReturnFn and FnsToEvaluate will be (...,$semiz$,$e1$,$e2$,$e3$,...). Note that if if you are not using a type of shock you simply omit it when writing the inputs. All three types of shocks can be made to depend on age (both the grid and transition probabilities can depend on age). When you use $semiz$ and e shocks you tell this to the toolkit by putting them in $vfoptions$ and $simoptions$.

A quick discussion of all of the different shock types/combos that we have seen in the Intro to Life-Cycle models. We saw in Life-Cycle models 8 and 9, an example with a single exogenous markov shock, z, first with a simple discrete markov process, and then with an AR(1) process that we discretized with the Farmer-Toda method. We saw in Life-Cycle model 11 an example with both an exogenous markov z and an exogenous i.i.d. e. Life-Cycle model 20 was an example using an exogenous markov z that dependend on age, and Life-Cycle model 21 does the same, but interpreting the shock as earnings productivity during working ages, and then reinterpreting as medical shocks during retirement. You can also find examples of discretizing some interesting earnings processes in Life-Cycle models 24 and 27; 24 is the 'standard' of a persistent shock, a transitory shock, and a fixed-effect, while 27 extends this to the 'state-of-the-art' with non-employment shocks and gaussian-mixtures. Life-Cycle model 28 was an example using two markov shocks and two i.i.d shocks, and having the shocks be correlated with each other. We saw in Life-Cycle model 29 an example with a semi-exogenous shock, and that semi-exogenous shock is age-dependent.

In this Appendix, first there are lots of examples that are just different versions of using a single markov exogenous shock z. These examples focus on how to turn different stochastic processes —an AR(1) with gaussian innovations, an AR(1) with gaussian-mixture innovations, a unit-root/random-walk— into discretized markov processes so that they can be used in the models; models A1, A2, A3. Then an example that show how to do age-dependent shocks, for both markov and i.i.d. exogenous states; model A4. These first four models, A1-A4, are all using the exact same model, just changing the exogenous shock processes.

Next we turn to how to do multiple shocks of a given type. There are four examples that demonstrate two markov shocks, $z1$ and $z2$; they first illustrate two independent shocks, and then show how to handle that the grids and/or transitions can depend on each other or even be correlated. The basic setup in VFI Toolkit is that with

two independent markovs you simply stack the columns for the two grids, but we can also switch to 'joint-grids', and these allow the values of the shocks to be correlated; models A5i, A5ii, A5iii, A5iv.[64] These four models, A5i-A5iv, are all using the exact same model, just changing the exogenous shock processes, and the only meaningful difference of this model from that used in A1-A4 is that there are now two markov exogenous states, rather than one.

Then an example with two i.i.d shocks, $e1$ and $e2$; models A6. To be sure that it is clear how to use lots of shocks, we then do an example with three markov with three i.i.d; model A7. We also show how to do two semi-exogenous shocks that are determined by two decision variables; model A8. Other than changing the number of shocks, and the types of shocks, we keep largely the same model as used in the previous examples

There are a few other types of shocks that might be of interest to discretize. One is VAR(1), vector autoregressions of lag-order 1, and we give an example discretizing a bi-variate VAR(1), which becomes two markov shocks using joint-grids so that the shock values and transitions can all be correlated; model A9. Mostly just out of interest, we also see how you can represent a second-order markov, by rewriting it as two first-order markovs, and we use this to discretize an AR(2); model A10.

The last example is just for understanding/interest. Model A11 shows how you can pretend that an i.i.d. shock is actually markov, and just have a transition matrix that has all the rows identical. This example is about helping you understand what the contents of the markov transition matrix are, it is not something you will ever want to do in practice as it is much faster to treat the i.i.d. shock as an actual i.i.d. shock, e.

A.1 List of Discretization methods in VFI Toolkit

Following is a brief list of the discretization methods present in VFI Toolkit. If you open the code for one of these at the top is a description of all the inputs, outputs, options, and it writes out the exact equation for the process being discretized.

These methods use four core approaches to discretize, so this paragraph attempts rough guide to how they set the grids and probabilities: Given a grid (typically evenly spaced points) Tauchen sets probabilities based on the conditional cdf, and Tanaka-Toda sets probabilites based on hitting the first few conditional moments. Tauchen-Hussey jointly chooses grids and probilities based on analytic solutions to hitting moments of an integral with respect to normal distribution. Rouwenhorst, reworks the probabilities to hit unconditional moments.

In all of the following, I recommend the Tanaka-Toda method as it tends to be the best performing these discretization methods.

To discretize an **AR(1) process with normally distributed innovations**

[64] Internally VFI Toolkit actually turns everthing into joint-grids and works solely with these. But being able to input stack columns is much easier and more intuitive for beginners, so the toolkit lets you work with these, and then just reformats them internally. The reason for this is that anything we can do with stacked columns can be done with joint-grids, but there are plenty of things with joint-grids that cannot be done with stacked columns. Hence joint-grids, while more complicated, are also much more powerful/featured. If you ever look at the internal code 'gridvals' is what the joint-grids get called, e.g. $z\_gridvals\_J$.

- *discretizeAR1_FarmerToda*: Tanaka-Toda method
- *discretizeAR1_Tauchen*: Tauchen method
- *discretizeAR1_TauchenHussey*: Tauchen-Hussey method
- *discretizeAR1_Rouwenhorst*: Tauchen method

Note, you can also use these to discretize a i.i.d. normal distribution, setting the autocorrelation to zero (and just use the first row of the transition matrix as your probabilities). 'Farmer-Toda' is the extension of Tanaka-Toda to markov processes (Tanaka-Toda was about i.i.d.).

To discretize an **AR(1) process with gaussian-mixture innovations**

- *discretizeAR1wGM_FarmerToda*: Tanaka-Toda method

To discretize an **AR(1) process with stochastic volatility**

- *discretizeAR1wSV_FarmerToda*: Tanaka-Toda method

To discretize an **VAR(1) process with normally distributed innovations**

- *discretizeVAR1_FarmerToda*: Tanaka-Toda method
- *discretizeVAR1_Tauchen*: Tauchen method

To deal with situations in which we have a life-cycle model where the exogenous shocks differ every period (whether because the parameters of the exogenous process vary with age, or because of a specific initial distiribution).

To discretize a **Life-Cycle AR(1) process with normally distributed innovations**

- *discretizeLifeCycleAR1_KFTT*: Tanaka-Toda method
- *discretizeLifeCycleAR1_FellaGallipoliPanTauchen*: Tauchen method
- *discretizeLifeCycleAR1_FellaGallipoliPan*: Rouwenhorst method

Note, these processes are sometimes also called 'non-stationary AR(1)' as they 'change' every period. Random-walks/Unit-roots can be discretized by this process, as they are just a specific type of non-stationary AR(1). 'KFTT' is the extension of Tanaka-Toda to life-cycle markov processes (Tanaka-Toda was about i.i.d.).

To discretize a **Life-Cycle AR(1) process with gaussian-mixture innovations**

- *discretizeLifeCycleAR1wGM_KFTT*: Tanaka-Toda method

Note, this is not an exhaustive collection of discretization methods. Others exist in the literature.

A.2 Life-Cycle Model A1: AR(1) process, alternative quadrature methods

We revisit Life-Cycle Model 9, which has AR(1) shocks that we discretized using the Farmer-Toda method (Farmer and Toda, 2017). Methods of discretizing shocks are known as 'quadrature methods'.[65] The Farmer-Toda quadrature method is just one approach to turning AR(1) processes into a discrete markov process (a grid and a transition matrix). In Life-Cycle Model A1 we show how to implement three alternatives: Rouwenhorst, Tauchen, and Tauchen-Hussey.[66] These alternatives have been widely used in the past, but the Farmer-Toda method performs better so I recommend you always use it, except when the autocorrelation coefficient of your AR(1) is ≥ 0.99 in which case Rouwenhorst method is recommended.[67]

Since there is no change in the model itself we will not repeat it here, see Life-Cycle Model 9. As a matter of good practice I recommend always checking that your results are not overly sensitive to your discretization (e.g., if you use 5 grid points, makes sure changing to 7 does not massively impact your results), you should also always describe in your paper which method you use to discretize shocks (so that other people can understand what you did; put it in the appendix, but it should be there somewhere).

A.3 Life-Cycle Model A2: AR(1) with gaussian-mixture innovations

The only change we make from Life-Cycle Model 9 is that instead of z being a AR(1) with iid normal innovations, it is an AR(1) with i.i.d. gaussian-mixture innovations; the gaussian distribution is just another work for normal distribution. Much like you can approximate real-valued continuous functions using polynomials, similarly you can approximate a wide range of probability distributions with a 'mixture' of normal distributions. So an AR(1) with gaussian mixture shocks is a way of modeling an AR(1) with complicated non-gaussian shocks, which are approximated with gaussian mixture shocks. We will use the method of Farmer and Toda (2017) to discretize it as a markov process. The rest of the model is unchanged so we do not report the whole model here. Instead we just report the equations for the AR(1) with gaussian mixture shocks.

A general AR(1) with non-gaussian shocks is given by,

$$x_t = (1-\rho)c + \rho x_{t-1} + \epsilon_t, \quad \epsilon_t \sim F$$

where the innovations ϵ_t are i.i.d. and their distribution is given by F.

[65] There are two general numerical methods to solve integrals (which we need to calculate expectations) using computers. The one we use is called 'quadrature', which involves evaluating the integral at a number of points (in our case the integrals are stochastic, so it is a number of grids points for the shock). The second is known as Monte Carlo integration and is never used for life-cycle models (you can find it being used for an economic model in Pál and Stachurski (2013), but quadrature is better for anything except high-dimensional shocks). There are essentially two aspects to discretizing AR(1) models to a markov process, the quadrature step of choosing grids, and a second step of choosing the transition matrix probabilities. Some quadrature methods treat these as seperate steps, like Farmer-Toda and Tauchen, while others treat them jointly, like Tauchen-Hussey and Rouwenhorst.

[66] Rouwenhorst (1995), Tauchen (1986) and Tauchen and Hussey (1991); see also Kopecky and Suen (2010) about the accuracy of Rouwenhorst being better for highly-persisent AR(1) processes.

[67] The basis for this recommendation are the results reported in Farmer and Toda (2017).

An AR(1) with 'gaussian mixture shocks' is exactly the same just with the addition that F is a gaussian mixture. We can have a gaussian mixture of n different normal distributions, but we will start with an example with a mixture of two normal distributions to get the idea. We have two normal distributions, $N(\mu_1, \sigma_1^2)$, and $N(\mu_2, \sigma_2^2)$, a guassian mixture has probability p_1 of being drawn from the first of these, and probability p_2 of being drawn from the second (obviously $p_1 + p_2 = 1$, so that the total probability is one). So we can write a guassian mixture of two normal (gaussian) distributions as, $F = p_1 N(\mu_1, \sigma_1^2) + p_2 N(\mu_2, \sigma_2^2)$. A general gaussian mixture of n normal distributions can be written as

$$F = \sum_{i=1}^{n} p_i N(\mu_i, \sigma_i^2)$$

You can estimate a gaussian mixture by standard methods like maximum likelihood or moment matching.

Notice that in the code to implement this we are going to have to provide the vector of $\{p_i\}_{i=1}^n$, the vector of $\{\mu_i\}_{i=1}^n$ and the vector of $\{\sigma_i\}_{i=1}^n$. The way the code works it will also require the 'unconditional mean', which is c in the above formula.[68]

We model F as a gaussian mixture. A gaussian mixture is flexible, so it can model a complicated F, yet analytically tractable, as all moments and the moment generating function have closed-form expressions. To give some intuition on why gaussian mixtures are useful, try to visualize the following $0.5N(2, 0.5) + 0.5N(0, 0.02)$, it will have a 'usual' normal distribution around 2, but also a sharp spike around 0, giving us a distribution with two peaks. To understand why they are convenient computationally the key is that the normal distributions is completely defined by just it's first two moments, the mean and variance; all the higher moments are just combinations of these. So when we add normal distributions together in a gaussian mixture everything just becomes weighted sums of these first two moments of each of the normal distributions we are mixing together.

We use the Farmer-Toda method to discretize the AR(1) with gaussian mixture shocks. The Rouwenhorst and Tauchen-Hussey methods cannot discretize this process as they are designed around just the first two moments. It is possible to extend the Tauchen method to AR(1) with gaussian mixture shocks, see Civales, Diez-Catalan, and Fazilet (2017), but VFI Toolkit does not presently contain an implementation of the Tauchen method for an AR(1) with gaussian-mixture innovations.[69]

A.4 Life-Cycle Model A3: Permanent (unit-root/random-walk) shocks

The only change we make from Life-Cycle Model 9 is that instead of z being a AR(1) with iid normal innovations, we instead have z follow a random-walk. We won't rewrite the value function problem itself, we just need to change the AR(1) process, $log(zprime) = \rho_z log(z) + \epsilon$, $\epsilon \sim N(0, \sigma_{\epsilon,z}^2)$, to instead be a (bounded) random walk process, $zprime = z + \epsilon_z$. Modelling permanent states typically requires more states for z to remain accurate and

[68] The code can actually discretize an AR(p) with gaussian mixture shocks, $(x_t - c) = \rho_1(x_{t-1} - c) + ... + \rho_p(x_{t-p} - c) + \epsilon_t$, $\epsilon_t \sim F$, and F is a gaussian mixture, $F = \sum_{i=1}^n p_i N(\mu_i, \sigma_i^2)$. It was originally written by Farmer and Toda (2017), please cite them if you use it.

[69] If anyone knows of code for this please let me know and I will adapt and add it.

we use 101, specifically $z\_grid = linspace(0.2, 2, 101)'$ (101 points equally spaced from 0.2 to 2, inclusive). We just need to set the transition matrix π_z to be a (bounded) random walk. We will choose ϵ_z to be have a probability of 0.5 of staying at the current value of z, a probability of 0.25 of going one grid point lower, and a probability of 0.25 of going one grid point higher. At the minimum grid point we will have a probability of 0.5 of staying, and a probability of 0.5 of going one point higher. At the maximum grid point we will have a probability of 0.5 of staying, and a probability of 0.5 of goind one point lower. Notice that this is a random walk, as $E[zprime|z] = z$, everywhere except at the maximum and minimum values, hence it acts as a random walk, except when it runs into the bounds.

The only things we need to change in the code are n_z, z_grid, and pi_z.

A.4.1 Special case: Permanent shocks with exogenous labor

There is a very nice way to implement permanent random walk shocks in models with exogenous labor that means way fewer grid points are required than if we just took the same naive approach as we did to add permanent shocks with endogenous labor in Life-Cycle Model A3. The key is in how the permanent shock enters the budget constraint. We can use a functional form for the shocks in the budget constraint that allows us to simply renormalize the model endogenous states in terms of the innovations in the permanent shock variable. We therefore only need a grid on z of the innovations, rather than the actual values of the permanent shock itself; e.g., for $y_t = y_{t-1} + \epsilon_t$ we would just need a grid on innovations ϵ, rather than a grid on y, and so we can get much higher accuracy from any given grid size.

Implementing this requires a substantial change to the model (no pensions, no labor productivity units as a deterministic function of age, etc.) so we will not attempt to explain the model here, you can find the full model both as a readable code and implementing codes at: discourse.vfitoolkit.com/t/permanent-shocks-to-income/127

The method and model are explained in full detail in Carroll (2021).

Note, the way we have written the code it would be trivial to make the innovations ϵ_t follow an AR(1) process.

Comment: The empirical evidence is pretty clear that earnings do not follow a unit root, e.g., Guvenen, Karahan, Ozkan, and Song (2021). So you probably shouldn't be doing this anyway. It is mainly here for historical/theoretical interest.

A.5 Life-Cycle Model A4: Age-dependent shocks

Empirical evidence shows that the (age-conditional) variance of income increases with age. One way to model this is to have shocks that increase in varance as age increases. If the shocks were transitory the age-conditional variance of consumption would not increase, because households would smooth consumption. So we also want these shocks to be persistent (so they imply a change in permanent-income and thus the life-cycle consumption hypothesis means consumption will also shift). How to model persistent shocks with a variance that increases with

age? We can model it as a 'non-stationary AR(1)' process, and discretize this using the 'generalized-Rouwenhorst' of Fella, Gallipoli, and Pan (2019). The model is exactly the same as that of Life-Cycle Model 9, except for changes to the exogenous shock process z on labor productivity units. So we will only describe the new z process here; see Life-Cycle Model 9 for the full model.

The 'non-stationary AR(1)' process for z is given by,

$$z_j = \rho_j z_{j-1} + \epsilon_j, \quad \epsilon_j \sim N(0, \sigma_{\epsilon,z,j})$$

notice the difference from a standard AR(1) because ρ_j and $\sigma_{\epsilon,z,j}$ both depend on age. We assume that $\rho_j < 1$, for all $j = 1, 2, ..., J$. Let, $\sigma_{z,j}$ be the standard deviation of z_j, then it follows that it evolves according to $\sigma^2_{z,j} = \rho^2_j \sigma^2_{z,j-1} + \sigma^2_{\epsilon,z,j}$. A full definition requires us to additionally specify z_0, which we do below.

Note that we enforce that the constant term is zero, so there is no 'drift' in the 'non-stationary AR(1)' process. In the original paper (Fella, Gallipoli, and Pan, 2019) implementing the discretization also require one further assumption, namely that $z_0 = 0$ (it has probability 1 of equaling zero). In the codes this is the default setting, but other initial distributions for z_0 can be implemented using the 'options' input.

Note that an extended Tauchen method also exists (it is in VFI Toolkit, but not used in any of these examples).

A.6 Life-Cycle Model A5i: Two markov exogenous states, z

We now consider models with two exogenous markov states, which we will call $z1$ and $z2$.

The household problem is,

$$\begin{aligned} V(a, z1, z2, j) \quad = \quad & \max_{c, aprime, h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\ & + s_j \beta E[V(aprime, z1prime, z2prime, j+1) | z1, z2] \\ & \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j z1 z2 h \\ & \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\ & 0 \leq h \leq 1, aprime \geq 0 \\ & log(z1prime) = \rho_{z2} log(z1) + \epsilon 1, \; \epsilon 1 \sim N(0, \sigma^2_{\epsilon,z1}) \\ & log(z2prime) = \rho_{z2} log(z2) + \epsilon 2, \; \epsilon 2 \sim N(0, \sigma^2_{\epsilon,z2}) \end{aligned}$$

Notice that $z1$ and $z2$ are independent of each other.

There are three changes we need to make to the codes to handle our two markov exogenous states, $z1$ and $z2$. First is just putting $(..., z1, z2, ...)$ as inputs to the ReturnFn and FnsToEvaluate. Second is to create z_grid. Third is to create pi_z.

The two shocks are independent, so we can just use the Tanaka-Toda method to discretize and AR(1) for each

of $z1$ and $z2$ separately, giving us $z1\_grid$, $z2\_grid$, pi_z1, and pi_z2.

We just combine the two grids as a stacked column vector, $z\_grid = [z1\_grid; z2\_grid]$ ($z1\_grid$ and $z2\_grid$ are both column vectors, and we are just 'stacking' them one on top of the other). Note that $size(z\_grid) = sum(n\_z) - by - 1$, this will always be true for stacked column vectors.

We also combine the two transition matrices, as $pi\_z = kron(pi\_z2, pi\_z1)$, taking the kronecker product, notice that they are in reverse order. Note that $size(pi\_z) = prod(n\_z) - by - prod(n\_z)$, which is always true of transition matrix across all possible setups.

A.7 Life-Cycle Model A5ii: Two markov (z) shocks, joint-grids

The model is completely unchanged from A5i. All that we will do is see an alternative way to rewrite z_grid, as a joint-grid instead of as a stacked column vector (which is what we did in A5i).

Think about what pi_z represents. Say $n\_z = [3, 2]$, $z1\_grid = [z1a; z1b; z1c]$, and $z2\_grid = [z2a; z2b]$. Then the rows and columns correspond the transitions from/to the combinations of $z1$ and $z2$ points given by,

$$\begin{bmatrix} z1a & z2a \\ z1b & z2a \\ z1c & z2a \\ z1a & z2b \\ z1b & z2b \\ z1c & z2b \end{bmatrix}$$

but if this is the 'grid' that pi_z is thinking about, we can just use it directly, and this is what we call a joint-grid for z.

So we set $z\_grid = [z1a, z2a; z1b, z2a; z1c, z2a; z1a, z2b; z1b, z2b; z1c, z2b]$. The interpretation is that the different rows index the possible combinations of $z1$ and $z2$ values, while the first column is all the $z1$ values and the second column is all the $z2$ values.

VFI Toolkit recognizes based on the size of the z_grid, which is now $prod(n\_z - by - length(n\_z)$ that this is a joint-grid and behaves appropriately.[70]

A.7.1 Age-Dependent Joint-Grids

When we saw age-dependent shocks in Life-Cycle model A4, we essentially added an extra dimension to the end of z_grid and pi_z of length N_j. The same thing applies for creating age-dependent shocks with joint-grids, we add model period as an extra dimension.

[70] Actually, internally VFI Toolkit converts everything into joint-grids and then works exclusively with joint-grids. Joint-grids are much more powerful/flexible in what kinds of things you can represent with joint-grids, as the next A5iii will make clear. But the stacked column vectors are much easier for users to setup and understand, as well as avoiding pointless duplication in the setup when the $z1$ and $z2$ grids are independent and we use the cross-product. So VFI Toolkit accepts the stacked column vectors, and internally just converts them to joint-grids and works with those. Internally the age-dependent joint-grids are referred to as $z\_gridvals\_J$, I called them gridvals instead of grid so I know they are joint-grids.

So since a joint-grid was $size(z\_grid) = prod(n\_z) - by - length(n\_z)$, we can set up an age-depenent joint-grid which will be $size(z\_grid\_J) = prod(n\_z) - by - length(n\_z) - by - N\_j$. Since $size(pi\_z) = prod(n\_z) - by - prod(n\_z)$ when using joint-grids, the age-dependent process will instead have $size(pi\_z\_J) = prod(n\_z) - by - prod(n\_z) - by - N\_j$. You just input these into commands as normal and based on their size VFI Toolkit understands to treat them as age-dependent shocks with joint-grids.

A.8 Life-Cycle Model A5iii: Joint-grids for Correlated Shocks, two markov z

The only change to the household problem from Life-Cycle Model A5iii is that we change the two markov processes $z1$ and $z2$, so we will not repeat the setup.

When we have two shocks that are correlated it makes sense to use a correlated joint-grid, as opposed to a cross-product grid (a.k.a. kronecker grid) as we have used until now. In a cross-product grid we choose, e.g., five points for $z1$ and three points for $z2$, and our grid on $(z1, z2)$ will be the cross-product of these, that is the 15 points we get from all combinations of the 5 points on $z1$ with the 3 points on $z2$. But when the shocks are correlated this may not make much sense as a grid. Imagine that the two shocks have a strong positive correlation. Then it is a waste to have a point on the grid which is a small value for $z1$ and a large value for $z2$, as this will essentially never occur; a cross-product grid when the shocks are highly correlated will have many points with essentially zero weight. We can avoid this problem by using a joint-grid, which in our present example would involve 25 points on $(z1, z2)$ chosen to reflect the correlation.

In terms of solving the model there is essentially no change. The shape of z_grid is now $prod(n\_z) - by - length(n\_z)$, as it standard for a joint-grid (correlated grids always take the form of joint-grids). VFI Toolkit recognises this size and automatically interprets it as a joint-grid and acts appropriately.

Obviously there is nothing stopping you from using jointly-determined grids when shocks are not correlated, just that there is no advantage from doing so. Life-Cycle Model A5ii was purely about helping you undertand what a joint-grid is.

A.9 Life-Cycle Model A5iv: Two markov (z) shocks, probabilities that depend on each other

The model is the same as in Life-Cycle model A5i, except that we change the two markov shocks, $z1$ and $z2$.

We will have two shocks each of which can take two values: z_1 which is employment/unemployment, and z_2 which is recession/expansion. The idea is that agents care about the microeconomics shocks (which directly effect them), but they do not directly care about the macroeconomic shocks (that only effect them indirectly). The macroeconomic shocks influence the microeconomic shocks (a recession makes unemployment more likely) which we model as the transition probabilities of the microeconomic shocks depending on the state of the macroeconomic shock.

So the grids of $z1$ and $z2$ are independent, and we set them up as a stacked column vector.

For the transition probabilities the macroeconomic shock z_2 will have 'its own' transition matrix, and then the microeconomic shock z_1 will have a transition probability that depends on z_2. See the code for how we do this, it is simple as long as you go slowly step-by-step in creating pi_z. We end up with a standard pi_z in the sense it has the same size and interpretation as always.

This modelling technique originates with Imrohoroglu (1989) - "Cost of business cycles with indivisibilities and liquidity constraints". For more 'advanced' versions of this idea you can use "rational inattention" so that households do not react to the macroeconomic variables. See, Maćkowiak and Wiederholt (2015) - "Business Cycle Dynamics under Rational Inattention." Or as life-cycle model in Carroll, Crawley, Slacalek, Tokuoka, and White (2020) - Sticky Expectations and Consumption Dynamics" (sticky expectations is a reduced-form way to model the concept of rational inattention).

A.10 Life-Cycle Model A6: Two i.i.d. (e) shocks

The model is the same as in Life-Cycle model A5i, except that we now use two i.i.d. shocks, $e1$ and $e2$, instead of two markovs.

There are three changes we need to make to the codes to handle our two i.i.d. exogenous states, $e1$ and $e2$. First is just putting $(..., e1, e2, ...)$ as inputs to the ReturnFn and FnsToEvaluate. Second is to create e_grid. Third is to create pi_e. Note the similarities to using two markovs.

The two shocks are independent, we will assume the first i.i.d. shock, $e1$ is normally distributed and use Tanaka-Toda to discretize it to get $e1\_grid$ and pi_e1. We assume the second i.i.d. shock, $e2$, is uniformally distributed and manually create $e2\_grid$ and pi_e2

We just combine the two grids as a stacked column vector, $e\_grid = [e1\_grid; e2\_grid]$ Note that $size(e\_grid) = sum(n\_e) - by - 1$, this will always be true for stacked column vectors.

We also combine the two probabilties (they are column vectors), as $pi\_e = kron(pi\_e2, pi\_e1)$, taking the kronecker product, notice that they are in reverse order. Note that $size(pi\_e) = prod(n\_z) - by - 1$, which is always true of probabilities for i.i.d. variables across all possible setups.

Note: we actually just reuse the $ReturnFn$ from Life-Cycle model A5i, since viewed from inside the return function there is no distinction between markov and i.i.d variables.

You can use joint-grids and/or age-dependence for i.i.d. variables, analagously to how it is done for markovs. See Life-Cycle model 27 for an examples with age-dependent i.i.d., and see Life-Cycle model 28 for an example where two i.i.d variables are correlated.

A.11 Life-Cycle Model A7: Three markov (z) and three iid (e) shocks

VFI Toolkit can handle up to four of each shock type, that is, each of *semiz*, z and e. Once you understand how to use two of a shock, using three is essentially a matter of combining the first two, and then combining the result of this with the third (in terms of creating grids and transition probabilties). This example is just to spell this out with an example code. This model is a bit silly, is just to clarify how to setup and use lots of shocks.

The household problem is,

$$\begin{aligned}
V(a,z1,z2,z3,e1,e2,e3,j) \quad = \quad & \max_{c,aprime,h} \frac{c^{1-\sigma}}{1-\sigma}-\psi\frac{h^{1+\eta}}{1+\eta}+(1-s_j)\beta\mathbb{I}_{(j>=Jr+10)}warmglow(aprime) \\
& +s_j\beta E[V(aprime,z1prime,z2prime,z3prime,e1prime,e2prime,e3prime,j+1)|z1, \\
& \text{if } j<Jr:\ c+aprime=(1+r)a+w\kappa_j z1z2z3e1e2e3h \\
& \text{if } j>=Jr:\ c+aprime=(1+r)a+pension \\
& 0\leq h\leq 1, aprime\geq 0 \\
& log(z1prime)=\rho_{z2}log(z1)+\epsilon 1,\ \epsilon 1\sim N(0,\sigma^2_{\epsilon,z1}) \\
& log(z2prime)=\rho_{z2}log(z2)+\epsilon 2,\ \epsilon 2\sim N(0,\sigma^2_{\epsilon,z2}) \\
& log(z3prime)=\rho_{z3}log(z3)+\epsilon 3,\ \epsilon 3\sim N(0,\sigma^2_{\epsilon,z3}) \\
& log(e1)\sim N(0,\sigma^2_{e,1}) \\
& e2\sim N(\mu_{e,2},\sigma^2_{e,2}) \\
& e3\sim U(e3_{min},e3_{max})
\end{aligned}$$

Notice that $z1$, $z2$ and $z3$ are independent of each other, and $e1$, $e2$ and $e3$ are similarly independent of each other.

As seen in the examples of Life-Cycle models A5i-A6, there are three keys to coding this. First is just putting $(...,z1,z2,z3,e1,e2,e3,...)$ as inputs to the ReturnFn and FnsToEvaluate. Second is to create *z_grid* and *e_grid*. Third is to create *pi_z* and *pi_e*.

With three shocks, we can easily set up the stacked column vector of the grids from each of the three shocks. For the transition matrix, just think of first combining two shocks (say $z1$ and $z2$) to get the transition matrix on these two (call it *pi_z1z2*), and then combine this with $z3$; so it is the same steps as for two shocks, just we repeat them a few times. Similarly for *pi_e*, we can first combine two shocks, and then combine the result of this with the third shock.

Note, you can do all of this with things like joint-grids, correlated shocks, and age-dependence.

A.12 Life-Cycle Model A8: VAR(1) persistent shocks

In Life-Cycle Model 10 we used two idiosyncratic shocks, z_1 and z_2, modelling one as an AR(1) and the other as i.i.d. Normal. It is possible to model two shocks as a VAR(1) and that is what we do here, discretizing them

using the Tauchen method for VAR(1). Since nothing changes in the model except the process on z_1 and z_2 we will only describe that here.

Let $z = [z_1; z_2]$, then we can write the bivariate VAR(1) as

$$z_t = \mu + \rho_z z_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \Sigma)$$

where μ is a 2-by-1 vector, ρ_z is a 2-by-2 matrix, ϵ_t is a 2-by-1 vector, and Σ is a 2-by-2 variance-covariance matrix.

We could write the same bivariate VAR(1) as the system of equations,

$$\begin{aligned} z_{1,t} &= \mu_1 + \rho_{z,1,1} z_{1,t-1} + \rho_{z,1,2} z_{2,t-1} + \epsilon_{1,t} \\ z_{2,t} &= \mu_2 + \rho_{z,2,1} z_{1,t-1} + \rho_{z,2,2} z_{2,t-1} + \epsilon_{2,t} \\ [\epsilon_{1,t}, \epsilon_{2,t}]' &\sim N(0, \Sigma) \end{aligned}$$

there relationship between the two being $\mu = [\mu_1; \mu_2]$ and $\rho_z = [\rho_{z,1,1}, \rho_{z,1,2}; \rho_{z,2,1}, \rho_{z,2,2}]$.

When using (our implementation of) the Tauchen method to discretize this VAR(1) we must also impose that Σ is diagonal, $\Sigma = [\sigma^2_{\epsilon,z,1}, 0; 0, \sigma^2_{\epsilon,z,2}]$.

VAR(1) with three or more variables can be implemented in the obvious manner. For more details see the discretization codes themselves.

In addition to the Tauchen method for discretizing VAR(1) there are also codes that implement the Farmer-Toda method for discretizing VAR(1), but these create 'jointly dependent' grids on z_1 and z_2 so cannot currently be used by VFI Toolkit.[71]

A.13 Life-Cycle Model A9: Two age-dependent semi-exogenous (semiz) shocks

How to use two semi-exgenous states, $semiz1$ and $semiz2$? This is actually already done in Life-Cycle model 29 so you can just go read that :D Instead this example is actually going to be about how to use more than one decision variable to influence the semi-exogenous states. The model with have two semi-exogenous states, and two decision variables that influence them. Essentially, you just use $vfoptions.numd\_semiz = 2$ to say that you want to use two decision variables to determine the semi-exogenous state transitions (default is 1). VFI Toolkit assumes that these are the 'last two' decision variables.

[71] We have been using a grid for z_1 and a grid for z_2, and the get a grid on $z = [z_1, z_2]$ by taking the kronecker product (cross product) of the two grids (the obvious way to create a grid on the two dimensions from two single dimension grids; this is all done internally by the codes so you won't see it). Farmer-Toda create a jointly-dependent grid on z, which is not the kronecker product of two seperate grids. Jointly-dependent grids are better, in principle, but require slightly different code to handle them and VFI Toolkit does not currently handle this.

Our household value function is now,

$$
\begin{aligned}
V(a,n,work,z,j) \quad = \quad & \max_{c,aprime,h,f,search} \frac{(c)^{1-\sigma}}{1-\sigma} + \eta_1 \frac{exp(j-\eta_3)}{(1+exp(j-\eta_3))}(\bar{n}+n)^{\eta_2} - search_c * search \\
& (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\
& +s_j \beta E[V(aprime, nprime, workprime, zprime, j+1)|z] \\
& \text{if } j < Jr: \ c + aprime = (1+r)a + work * w\kappa_j zh - childcarecosts + (1-work)benefits \\
& \text{if } j >= Jr: \ c + aprime = (1+r)a + pension \\
& childcarecosts = childcarec(h>0)n \\
& 0 \leq h \leq 1, aprime \geq 0 \\
& log(zprime) = \rho_z log(z) + \epsilon, \ \epsilon \sim N(0, \sigma^2_{\epsilon,z})
\end{aligned}
$$

the idea is that fertility, f determines number of children n (similar to in Life-Cycle Model 29, but with a less detail), and *search* means you are looking for a job and determines *work* which is whether or not you have a job (binary 1 for job, 0 for no job). If you don't have any work you just receive benefits.

Note that there are two semi-exogenous states, n and *work*, and these are controlled by two decision variables, f and *search*. VFI Toolkit handles this just using one 'SemiExoStateFn' that determines how the semi-exogenous states evolve given the two decisions. As mentioned earlier we also need to use $vfoptions.numd\_semiz = 2$ to specify that it is the 'last two' decision variables that determine the transitions of the semi-exogenous states (by default on the last decision variable would be used).

The setup for the semi-exogenous states is that: If you have work=1, then you lose your job with an exogenous job-seperation probability, and otherwise keep the job. If you have work=0, then you find a job with a probability of 'search' (so your search effort directly determines your probability of finding a job). Children become adults with a certain probability, and if you choose f=1, then you will have a child with a certain probability.

This kind of setup with work being semi-exogenous and depending on the effort expended in job search is called a 'search-labor' model (not to be confused with 'search-and-matching', which is an equilibrium concept).

A.14 Life-Cycle Model A10: Second-Order Markov Processes (implementing an AR(2) persistent shock)

The VFI Toolkit can only handle first-order Markov processes, which are typically just referred to as markov processes. But fortunately you can rewrite any second-order Markov process with one state as a first-order Markov process with two states (the original state and the lag of that state). Using second-order Markovs processes is rare in economic models, but let's see how.

Before explaing how to convert the second-order Markov processes into a first-order Markov process we explain the details of how to change a Markov process with two variables into a single (vector valued) variable Markov

process. We have already done this in the Life-Cycle Models that contain two exogenous shocks (such as Life-Cycle Model 11), but it will help to see it formally.

How to turn a Markov process with two variables into a single variable Markov process:
Say you have a Markov process with two variables, z & y, we can treat them as a single vector-valued markov process $x = (z, y)$. Let z take the states $z_1, ...z_n$, and y take the states $y_1, ..., y_m$. We start with the simple example of how to combine the two when their transitions are completely independent of each other so as to illustrate the concepts and then treat the general case.

When z and y are independent Markov processs, $Pr(z_{t+1} = z_j) = Pr(z_{t+1} = z_i|z_t = z_j) = \pi^z_{ij}$ $\forall i, j = 1, ...n$ and $Pr(y_{t+1} = y_j) = Pr(y_{t+1} = y_i|y_t = y_j) = \pi^y_{ij}$ $\forall i, j = 1, ...m$, then we can define a new single-variable Markov process x simply by taking the Kronecker Product of z and y. Thus, x will have n times m states, $[x_1, .., x_{nm}]' = [z_1, ...z_n]' \otimes [y_1, ...y_m]'$, and it's transition matrix will be the Kronecker Product of their transition matrices; $\Pi^x = \Pi^z \otimes \Pi^y$. For the definition of the Kronecker product of two matrices see wikipedia. Note that in the codes you only need to combine the transition matrices to create the joint transition matrix; you simply include the grids as a 'stacked' column vector and the rest is done internally by VFI Toolkit.

Note that we can let z and y depend on each other (as in the case that z and y form a bivariate VAR(1), treated in Life-Cycle Model A7), although obviously the transition matrix becomes more complex. Another case of potential interest is when one of z and y is 'microeconomic' and the other is 'macroeconomic', which we capture by allowing the microeconomic to depend on the macroeconomic, but not vice-versa. Thus we assume, without loss of generality, that z_t is the microeconomic variable and evolves according to the transformation matrix defined by $Pr(z_t|z_{t-1}, y_t, y_{t-1})$, while y_t has the transformation matrix $Pr(y_t) = Pr(y_t|y_{t-1})$[72]. The transition matrix however is now more complicated; rather than provide a general formula you are referred to the example in Life-Cycle Model A8.

If you have three or more variables in vector of the first-order Markov process you can reduce this to a scalar first-order Markov process simply by iteratively combining pairs. For example with three variables, (z^1, z^2, z^3), start by first defining x^1 as the combination of z^1 & z^2 (combining them as described above), and then x as the combination of x^1 and z^3.

How to turn a second-order Markov process into a first-order Markov process:
Suppose that z, with states $z_1, ...z_n$ is a second-order Markov process, that is $Pr(z_{t+1} = z_i) = Pr(z_{t+1} = z_i|z_t = z_j, z_{t-1} = z_k) \neq Pr(z_{t+1} = z_i|z_t = z_j)$. Consider now the vector (z_{t+1}, z_t). It turns out that this vector is in fact a vector first-order Markov process (define this periods state (z_{t+1}, z_t) in terms of last periods state (z_t, z_{t-1}) by defining this periods z_{t+1} as a function of last periods z_t & z_{t-1} following the original second-order Markov process, and define this periods z_t as being last periods z_t). Now we have a 'vector-valued' first-order Markov process on the vector (z_t, z_{t-1}).

Obviously this idea can be trivially extended to turn, e.g., a third-order Markov process into a first-order

[72]Note that here I switch from describing Markov processs as $t + 1|t$ to $t|t - 1$, the difference is purely cosmetic.

Markov process with three states.

Life-Cycle Model A4: idiosyncratic shock, AR(2) We will provide an example that takes Life-Cycle Model 9 and makes the sole change of switching from an AR(1) process for the exogenous shock on (log) labor productivity units z to instead being an AR(2) process.

$$z_t = \rho_{z,1} z_{t-1} + \rho_{z,2} z_{t-2}$$

We can use the Farmer-Toda method to discretize this, the command we use is actually for a more general AR(p) process with gaussian mixture innovations, but we will just use an AR(2) with gaussian innovations (we will have to code it as a 'mixture of one normal').

The Farmer-Toda method creates a first-order Markov process on (z_t, z_{t-1}), which we will refer to in the codes as (z_1, z_2). Note that while both z_1 and z_2 are exogenous state variables the return function will only actually use the current value z_1 (which represents z_t); both have to be passed as inputs to the return function as they are both states, but only one ends up being used in the return function (the other sill matters for things like expectations).

A.15 Life-Cycle model A11: Model an i.i.d. as an markov exogenous state

The model is just that of Life-Cycle Model 11. Except that we will model the i.i.d exogenous state 'e' by treating it like a second markov exogenous state, 'z2'. This is a bad idea in the sense it will make all the codes slower and is not something you will ever want to do in practice. It is done here with the hope of helping you better understand how the transition matrix of a markov state works. Specifically, very row of the transition matrix for a markov shock can be though of as the i.i.d distribution of next period values conditional on this period value.[73]

The persistent shocks, z_1, we will model as an AR(1). The transitory shock, z_2 we will model as i.i.d.

Our household value function now has $z_1 z_2$ in place of z to determine idiosyncratic producitivity units,

$$\begin{aligned}
V(a, z_1, z_2, j) \quad = \quad & \max_{c, aprime, h} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} + (1-s_j)\beta \mathbb{I}_{(j>=Jr+10)} warmglow(aprime) \\
& \qquad + s_j \beta E[V(aprime, z_1 prime, z_2 prime, j+1) | z_1] \\
& \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j z_1 z_2 h \\
& \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\
& 0 \leq h \leq 1, aprime \geq 0 \\
& log(z_1 prime) = \rho_{z1} log(z_1) + \epsilon, \; \epsilon \sim N(0, \sigma^2_{\epsilon, z1}) \\
& log(z_2) \sim N(0, \sigma^2_{z2})
\end{aligned}$$

[73] E.g., think about the AR(1) process $z_t = \rho_z z_{t-1} + \epsilon_t$, and rewrite $z_t - \rho_z z_{t-1} = \epsilon_t$. Notice that the left side is the 'conditional distribution', that is the change in z, and the right side is just the i.i.d. innovation. This is what the rows of the markov transition matrix represent, with each different row corresponding to a different value of z_{t-1}.

Notice that z_1 is AR(1) and z_2 is i.i.d. normal (in logs).

Note that to get an exogenous shock (log) z_2 which is i.i.d. $N(0, \sigma_{z2}^2)$, we can simply use the same method as we did for the AR(1) process and just set $\rho = 0$.[74]

How do we let the codes know we have two exogenous shocks? First we have to set n_z to be a vector with the number of grid points of the first and second exogenous shocks, so if we use 21 points for the AR(1) shock and 5 for the iid shock we will set $n\_z = [21, 5]$. Second we set $z\_grid = [z1\_grid; z2\_grid]$; (we 'stack' the column grids of the two shocks). Third, we set the joint transition matrix $pi\_z = kron(pi\_z2, pi\_z1);$, by using the kronecker product, notice that when doing this you must put them in reverse order. We also need to modify the inputs to the ReturnFn and FnsToEvaluate so that they include both $z1$ and $z2$.

[74] If we wanted any other i.i.d. distribution we just create the transition matrix for z_2, π_{z2}, so that it has all rows with that same distribution. Recall that a row of the transition matrix represents the probabilities of going from the state that the row represents to each of the other states, so if all the rows are the same it is saying that the current state does not matter, which means it will be i.i.d. So, e.g., we could make z_2 a uniformly distributed variable by making the transition matrix a matrix of ones, divided by the number of states of z_2 (really we just want to divide each row, but dividing the whole matrix is an easy way of implementing this).

B Behavioural Preferences

This appendix provides more explanation both about how some of the non-standard life-cycle models work conceptually, as well as about how they should be used in the codes.

B.1 Epstein-Zin preferences

This appendix covers some details and options when using Epstein-Zin preferences. The main concepts are that you can do Epstein-Zin in consumption units (here called 'traditional') or in utility utils, and that the way in which bequests are handled when using Epstein-Zin preferences has to be done with care.

The point of Epstein-Zin preferences is that under standard recursive vonNeumann-Morgenstern expected utility there is just one parameter that is determining both the risk aversion and the elasticity of intertemporal substitution, which are importantly distinct concepts. See section B.1.3 for an explanation of these concepts.

Using Epstein-Zin preferences with a utility function is as simple as setting *vfoptions.exoticpreferences='EpsteinZin'*, and specifying the name of the parameter that influences the risk aversion, *vfoptions.EZriskaversion*. Then saying whether the utility function is positive valued ($u \geq 0$) or negative valued ($u < 0$); negative valued is *vfoptions.EZpositiveutility=0*. Note that the interpretation of the risk aversion parameter depends on whether the utility function is positive or negative valued.

When using Epstein-Zin preferences the conditional survival probabilities can no longer just be treated like discount factors. You therefore need to specify them using *vfoptions.survivalprobability*.

If you plan to use warm-glow of bequests these have to be treated specially. This is explained in Appendix B.1.5.

Notice that once you have solved the value function problem to get the optimal policy function everything else about a heterogeneous agent model depends only on the optimal policy and so we don't need to take account of Epstein-Zin preferences for things like simulating the model. The exception is any kind of welfare evaluation.

There are two alternatives for how to set up Epstein-Zin preferences. Epstein-Zin in utility units, and the traditional Epstein-Zin in consumption units.

B.1.1 Epstein-Zin with Utility function

We start with the utility function version. With standard recursive vonNeumann-Morgenstern expected utility we have a value function problem,

$$V(a,z,j) \quad = \quad \max_{d,aprime} u(d,aprime,a,z) + \beta E[V(aprime,zprime,j+1)|z]$$

with some constraints that I omit here as they are not relevant to the concepts around Epstein-Zin preferences.

Epstein-Zin preferences modify this to,

$$V(a,z,j) \quad = \quad \max_{d,aprime} u(d,aprime,a,z) + \beta E[V(aprime,zprime,j+1)^{1-\phi}|z]^{\frac{1}{1-\phi}}$$

where ϕ is any real number. Notice that this is the same as the standard case, except that the value function is now 'twisted' and 'untwisted' by the coefficient $1-\phi$. When $\phi = 0$ this just reduces to the standard case.

This formulation works for $u > 0$. If instead $u < 0$, then we need to formulate the Epstein-Zin preferences slightly differently as,

$$V(a,z,j) \quad = \quad \max_{d,aprime} u(d,aprime,a,z) - \beta E[-V(aprime,zprime,j+1)^{1+\phi}|z]^{\frac{1}{1+\phi}}$$

where again ϕ is any real number, and $\phi = 0$ reduces to the standard case.[75]

A higher ϕ corresponds to a greater degree of risk aversion ($\phi > 0$ means that the preferences are more risk averse that vonNeumann-Morgenstern preferences would be).[76]

When using *vfoptions.exoticpreferences='EpsteinZin'* by default the codes assume *vfoptions.EZutils=1*, which is that the Epstein-Zin preferences are in utility units. You do still need to specifty *vfoptions.EZpositiveutility*.

B.1.2 'Traditional' Epstein-Zin with consumption units

If you set *vfoptions.EZutils=0* you get the 'traditional' Epstein-Zin preferences used by Epstein and Zin (1989) and many others. These measure the value function in terms of consumption streams rather than utils, and so are also refered to as Epstein-Zin preferences in consumption units.

The 'traditional' setup when using Epstein-Zin preferences in VFI Toolkit is to solve the value function problem,

$$V(a,z,j) \quad = \quad \max_{c,aprime} \left[u^{1-1/\psi} + \beta (E[V(aprime,zprime,j+1)^{1-\gamma}|z])^{\frac{1-1/\psi}{1-\gamma}} \right]^{\frac{1}{1-1/\psi}}$$

with some constraints that I omit here as they are not relevant to the concepts around Epstein-Zin preferences. The return function for a model with exogenous labor is just $u = c$. If we have endogenous labor it will be different, namely $c^{1-\chi}(1-h)^{\chi}$.

In the traditional formulation of Epstein-Zin preferences we have γ which determines the risk aversion, and a seperate parameter ψ which is the elasticity of intertemporal substitution.[77] Epstein-Zin preferences imply that the agents have preferences for early or later resolution of uncertainty. If $\gamma > 1/\psi$ they prefer an early resolution

[75] Because of how the codes work it is not possible to let $u = 0$. To fit with standard ways in which the return function is used in VFI Toolkit in both cases you can also set the return function to $-\infty$, for example to rule out certain possibilities.

[76] This is why we use $1-\phi$ when $u > 0$ and $1+\phi$ when $u < 0$. This way the interpretation of ϕ is the same in both cases.

[77] The original paper of Epstein and Zin (1989) uses ρ and α as the parameters. Relating to the present formulation, $\rho = 1 - 1/\psi$ and $\alpha = 1 - \gamma$. The nicety of the formulation here is that ψ and γ relate directly to the elasticity of intertemporal substitution and risk aversion, and that we need both parameters to be greater than zero (whereas the original EZ parameters have to be less than one).

of uncertainty, and if $\gamma < 1/\psi$, they prefer a later resolution of uncertainty.

This formulation of Epstein-Zin preferences is known as Epstein-Zin in 'consumption units' (as $u = c$). It is probably the most common formulation. But it lacks one nice property, namely it does not really nest standard vonNeumann-Morgenstern preferences with CRRA utility ($u(c) = \frac{c^{1-\gamma}}{1-\gamma}$) as it is missing the $1/(1-\gamma)$.[78] This is why we also have the utility units formulation of Epstein-Zin preferences described previously.

If you want to use Epstein-Zin preferences with consumption units just set *vfoptions.EZutils=0* and *vfoptions.exoticpreferences='EpsteinZin'*. Then you need to specify the names of the risk aversion and elasticity of intertemporal substition parameters, γ and ψ in the above equations, using *vfoptions.EZriskaversion* and *vfoptions.EZeis*.

Sometimes people want a $(1-\beta)$ term multiplying the period-return.[79] You can do this by setting *vfoptions.EZoneminusbeta=1*, which will use the discount factor to add the appropriate term here.

B.1.3 Epstein-Zin preferences: risk aversion and elasticity of intertemporal substitution

In standard economic models agents expectations about the future are based on 'von-Neumann-Morgenstern' expected utility. This involves two main aspects, preferences are time-seperable, and the future utilities matter in terms of their expectation. An unintentional side-effect of this is that both risk aversion and the intertemporal elasticity of consumption get determined by the same parameter. So for example in a standard two-period model of consumption-savings decision the 'value' in the first period is[80]

$$u(c_1) + \beta E[u(c_2)] \tag{1}$$

or if we were to use the constant-elasticity-of-substitution (CES) utility function,

$$\frac{c_1^{1-\gamma}}{1-\gamma} + \beta E\left[\frac{c_2^{1-\gamma}}{1-\gamma}\right] \tag{2}$$

where γ will determine both the risk aversion and the intertemporal elasticity of substitution. $\frac{-cu''(c)}{u'(c)} = \gamma$ is the relative risk aversion (RRA), the CES utility function displays constant RRA, called CRRA.[81] $ln\left(\frac{u'(c_{t+1})}{u'(c_t)}\right) = 1/\gamma$ is intertemporal elasticity of substitution. Notice how just one parameter γ determines both the risk aversion and

[78] The traditional EZ preferences do nest vonNeumann-Morgenstern preferences when $\psi = 1/\gamma$, but the resulting utility function is an oddly written one. You instead get $V = \max_c\{c^{1-\gamma} + \beta E[V'^{1-\gamma}]\}^{1-\gamma}$, to see that this is CRRA imagine starting in the last period, you get $c^{1-\gamma}$ and then put this to the inverse power, when you put it into the previous period value function you get the power back from the $V'^{1-\gamma}$. So this is CRRA, but the same value of γ is not going to deliver the same preferences as if we put the CRRA utility into a standard value function with vonNeumann-Morgenstern risk preferences.

[79] They want $(1-\beta)u^{1-1/\psi}$ instead of $u^{1-1/\psi}$. I am not sure why people do this, but I have seen it in papers. If you know why please let me know and I will add explanation here.

[80] To keep the notation easier I will only describe the utility functions, and skip over the budget constraints and the like.

[81] Actually this is not really the correct way to measure risk-aversion in models with many time periods and/or many goods, see Swanson (2012), and it becomes more subtle again when using Epstein-Zin preferences (Swanson, 2018).

the intertemporal elasticity of substitution.

(Traditional) Epstein-Zin preferences (Epstein and Zin, 1989) seperate the risk aversion from the intertemporal elasticity of substitution, using a different parameter to determine each of these. In our simple two-period model using Epstein-Zin preferences the 'value' in the first period becomes

$$\left[c_1^{1-1/\psi}+\beta\left(E\left[(c_2^{1-1/\psi})^{1-\gamma}\right]\right)^{\frac{1-1/\psi}{1-\gamma}}\right]^{\frac{1}{1-1/\psi}}$$

where now γ is the CRRA and ψ is the intertemporal elasticity of substitution. It is important to note that we cannot write this in the form of von-Neumann-Morgenstern expected utility preferences as in (1). Except for the case when $1/\psi=\gamma$, in which case both play the role of γ in equation (2); note that with $1/\psi=\gamma$ the Epstein-Zin preferences will be vNM preferences, but they do not just reduce to CES with vNM.

Epstein-Zin preferences with a utility function have a stardard setup in terms of the intertemporal elasticity of substitution (which will depend on what utility function you use), and then makes risk aversion different to this by increasing/reducing the amount of risk aversion using the parameter ϕ in the earlier formulation.

Both the utility units and the (traditional) consumption units formulations of Epstein-Zin preferences nest von-Neumann Morgernstern as a special case. Specifically, if we use the utility units Epstein-Zin preferences with $\phi=0$ then it simplifies to being a CES utility with von-Neumann Morgenstern. If we use the (traditional) consumption units Epstein-Zin preferences with $\gamma=1/\phi$ then we get the same preferences as a CES utility with von-Neumann Morgernstern preferences, except that the units are consumption units instead of utils.[82] This was used to test the implemenations: that Epstein-Zin with utility units, Epstein-Zin with consumption units, and CES utility with von-Neumann Morgenstern all give the same solution for the policy function (both for the case of exogenous labor and for the case of endogenous labor); the codes for these six cases are here.

B.1.4 Conditional Survival Probabilities with Epstein-Zin preferences

Conditional survival probabilities are a risk, while the discount factor is about time-preference. Since Epstein-Zin is about seperating risk from intertemporal substition unsurprisingly we now also need to treat conditional survival probabilities differently from the discount factor. We simply specify name of the parameter for the conditional survival probabilities using *vfoptions.survivalprobability* (e.g., *vfoptions.survivalprobability='sj'*).

The conditional survival probabilities appear multiplying next period value function inside the expectation operator. So for example in Epstein-Zin in utility units with positive-valued utility function it will be,

$$V(a,z,j) \quad = \quad \max_{d,aprime} u(d,aprime,a,z)+\beta E[s_j V(aprime,zprime,j+1)^{1-\phi}|z]^{\frac{1}{1-\phi}}$$

[82] Utility is only unique up to an affine transformation (if you take any utility function, multiply by a positive constant, and then add any constant, you just get the same preferences). So the consumption units give the same policy, but different value function.

for Epstein-Zin in utility units with negative-valued utility function it will be,

$$V(a,z,j) = \max_{d,aprime} u(d,aprime,a,z) - \beta E[-s_j V(aprime,zprime,j+1)^{1+\phi}|z]^{\frac{1}{1+\phi}}$$

while for Epstein-Zin in consumption units it will be

$$V(a,z,j) = \max_{c,aprime} \left[u^{1-1/\psi} + \beta(E[s_j V(aprime,zprime,j+1)^{1-\gamma}|z])^{\frac{1-1/\psi}{1-\gamma}}\right]^{\frac{1}{1-1/\psi}}$$

Notice how in all three of these cases it is not possible to simply pull s_j out of the expectation next to β (which can be done with standard vonNeumann-Morgenstern risk preferences).

Using Epstein-Zin preferences instead of standard preferences has important advantages around the 'value of life' in life-cycle models (Cordoba and Ripoll, 2017). Note that if you are looking at the value of life, then if death is associated with zero utility you clearly want to use a positive-valued utility function with the Epstein-Zin preferences so that life is preferred to death.

Note, how exactly survival probabilities should be treated in Epstein-Zin preferences remains controversial. If you do want the conditional surivival probability to be treated as just another discount factor, you can do this by just not specifying *vfoptions.survivalprobability*, and instead including it in the *DiscountFactorParamNames*.

For this reason the codes also allow another possible formulation, suggested by Cordoba and Ripoll (2017), which essentially uses Epstein-Zin twice, once for the uncertainty, and once for the conditional survival probabilities, with a different parameter for each. Epstein-Zin in utility units with positive-valued utility function it will be,

$$\begin{aligned} V(a,z,j) = & \max_{d,aprime} u(d,aprime,a,z) + \\ & \beta[s_j \left(E[V(aprime,zprime,j+1)^{1-\phi}|z]^{\frac{1}{1-\phi}}\right)^{1-\phi_m} + (1-s_j)W(aprime)^{1-\phi_m}]^{\frac{1}{1-\phi_m}} \end{aligned}$$

for Epstein-Zin in utility units with negative-valued utility function it will be,

$$\begin{aligned} V(a,z,j) = & \max_{d,aprime} u(d,aprime,a,z) \\ & -\beta[-s_j(-1)\left(E[-V(aprime,zprime,j+1)^{1+\phi}|z]^{\frac{1}{1+\phi}}\right)^{1+\phi_m} - (1-s_j)W(prime)^{1+\phi_m}]^{\frac{1}{1+\phi_m}} \end{aligned}$$

while for Epstein-Zin in consumption units it will be

$$\begin{aligned} V(a,z,j) = & \max_{c,aprime} [u^{1-1/\psi} + \\ & \beta[s_j\left(E[V(aprime,zprime,j+1)^{1-\gamma}|z]^{\frac{1}{1-\gamma}}\right)^{1-\gamma_m} + (1-s_j)W(aprime)^{1-\gamma_m}]^{\frac{1-1/\psi}{1-\gamma_m}}]^{\frac{1}{1-1/\psi}} \end{aligned}$$

and this can be implemented by setting *vfoptions.EZmortalityriskaversion* to specify the name of the parameter here denoted ϕ_m. $W(aprime)$ is the utility on death (so is just the same as using the warm-glow of bequests as explained in the next section); if you do not decare it then by default the codes will use $W(aprime) = 0$. Importantly, people display a preference for early resolution of financial risk, and delayed resolution of survival risk, and this formulation permits both of these to occur (under certain paramterizations).

B.1.5 Bequests with Epstein-Zin preferences

Bequests with Epstein-Zin preferences are subtle, and so VFI Toolkit has been coded to deal with this for you. Kraft, Munk, and Weiss (2022) show that if you naively use warm-glow of bequests in a model with Epstein-Zin preferences then it just leads to garbage (the parameters are not related to the strength of the bequest motive).

The key to seeing why and how bequests have to be treated carefully in Epstein-Zin preferences is the expectations term, $\beta E[V(aprime, zprime, j+1)^{\alpha}|z]^{\frac{1}{\alpha}}$, where I have made the powers a generic α and 1α so that it is clearer that this covers both the utility-units and consumption-units cases. Notice that adding in bequests, which occur next period, cannot simply be added after the end of this term, as they would be in standard von-Neumann Morgenstern risk preferences. This is because the next period warm-glow is going to need to be raised to the power of α, then added into next period expectation, and then all raised to $1/\alpha$. That is, the next period term will be $\beta E[s_j V(aprime, zprime, j+1)^{\alpha} + (1-s_j)W(aprime)^{\alpha}|z]^{\frac{1}{\alpha}}$, where $W(aprime)$ is the warm-glow of bequests.

Implementing this in the codes therefore requires to things, first we have to specify the conditional survival probability, *vfoptions.survivalprobability='sj'*, where 'sj' is the name of a (likely age dependent) parameter stored in the standard structure of parameters (called Params in most examples). Second, we need to specify the warm-glow function $W(aprime)$. We do this by setting up *vfoptions.WarmGlowBequestsFn* which depends on next-period endogenous state (on *aprime*). This just leaves the question of what warm-glow function is appropriate? The key is just to use the same function as our regular utility function. So if our utility of consumption is $\frac{c^{1-\sigma}}{1-\sigma}$, then we use the warm-glow of bequests function $\frac{aprime^{1-\sigma}}{1-\sigma}$. We multiply this by a constant wg to determine the importance of bequests. Hence our final warm-glow of bequest function is $wg\frac{aprime^{1-\sigma}}{1-\sigma}$.[83] If by contrast we use Epstein-Zin preferences in consumption-units with exogenous labor, then we had the return function being $u = c$, and so our warm-glow of bequests function would be c/wg, where again wg is a constant that determines the strength of the bequest motive (notice that once this is raised to $1 - 1\psi$, dividing by wg is moving us to the region of the utility function where marginal utility of consumption is high, and changes fast).

wg controls the strength of the bequest motive, which is monotone in wg. Note that this is not the precise specification of Kraft, Munk, and Weiss (2022) who wish to impose a precise interpretation on the parameter wg;

[83] An alternative would be to use $W(aprime) = \frac{(aprime/wg)^{1-\sigma}}{1-\sigma}$. This just changes the interpretation of wg and exactly how it relates to bequests (in both cases the size of the bequest will be monotone in wg. This —different specifications lead to subtly different interpretations of wg— is the focus of Kraft, Munk, and Weiss (2022) who have a very specific interpretation in mind.

you can get their specification by setting *vfoptions.WarmGlowBequestsFn* appropriately.[84] Note that while this is the warm-glow of bequest because it is 'next period' you have to use Epstein-Zin to bring it into the final period which is what the VFI Toolkit is doing when you use *vfoptions.WarmGlowBequestsFn*.

Note that because we specify the conditional survival probability seperately, you do not include it in DiscountFactorParamNames (if you do it will just be used in the same way as β in the Epstein-Zin codes).

If you using the utility utils formulation of Epstein-Zin preferences with a positive utility function then the warm-glow-of-bequests works like,

$$V(a,z,j) = \max_{d,aprime} u(d,aprime,a,z) + \beta E[s_j V(aprime,zprime,j+1)^{1-\phi} + (1-s_j)W(aprime)^{1-\phi}|z]^{\frac{1}{1-\phi}}$$

If instead $u<0$, then warm-glow of bequests with Epstein-Zin preferences is,

$$V(a,z,j) = \max_{d,aprime} u(d,aprime,a,z) - \beta E[(-s_j V(aprime,zprime,j+1))^{1-\phi} - (1-s_j)W(aprime)^{1-\phi}|z]^{\frac{1}{1-\phi}}$$

If you have Epstein-Zin in consumption-units when using warm-glow-of-bequests the problem being solved is,

$$V(a,z,j) = \max_{c,aprime} \left[u^{1-1/\psi} + \beta E[s_j V(aprime,zprime,j+1)^{1-\gamma} + (1-s_j)W(aprime)^{1-\gamma}|z]^{\frac{1-1/\psi}{1-\gamma}}\right]^{\frac{1}{1-1/\psi}}$$

So with consumption-units and exogenous labor you want *vfoptions.WarmGlowBequestsFn*$= wgaprime^{1-1/\psi}$, and with consumption-units and endogenous labor you want *vfoptions.WarmGlowBequestsFn*$= wg(aprime^{1-\chi}(1-\bar{h})^{\chi})^{1-1/\psi}$, where $\bar{h}$ is a constant that denotes the 'typical/reference' hours worked (if no-one works in retirement, you will way $\bar{h}=0$).

Note that if you use a setup where *aprime* is uncertain, like using *Case3* to solve portfolio-choice models, then this just involves changing $warmglow^{\alpha}$ to $E_u[warmglow^{\alpha}]$ in the general formulation at the beginning of this section on bequests, and this is automatically done by the codes.

Summing up, using warm-glow-of-bequests in a life-cycle model with Epstein-Zin preferences is as simple as correctly declaring *vfoptions.WarmGlowBequestsFn*.

[84] To implement the Kraft, Munk, and Weiss (2022) specification set the warm-glow of bequest function when using consumption units to $wg^{\frac{1}{\psi-1}}aprime$. In the Kraft, Munk, and Weiss (2022) specification for the warm-glow of bequests, wg (they denote it epsilon) can be interpreted as the target ratio of terminal wealth to terminal consumption (terminal as in end of the final period J). This is precisely true in simple settings and an accurate approximation in more advanced settings. In simple settings the ratio of bequest wealth to total lifetime consumption will equal $(wg+J)/J$. This remains roughly true but only as a mediocre approximation in more advanced settings. They also show that if you want to use a CES warm-glow of bequests in a regular setting with CES preferences (no Epstein-Zin) then you should use $warmglow = \frac{1}{1-\gamma}wg^{1-\gamma}aprime^{1-\gamma}$ to get their precise interpretation of wg as measuring the relative weight on bequests (most papers would use wg instead of $wg^{1-\gamma}$).

B.2 Quasi-Hyperbolic Discounting

Quasi-hyperbolic preferences are a tractable way to model 'impatience', with the present-self taking decisions that are not in the interest of their own future-self. The standard way to discount the future is exponential discounting, which uses the same discount factor β between any two consecutive periods. Quasi-hyperbolic discounting involves two discount factors, β_0 which is used as the additional discount factor between the current period and the next period, and β which is used as the discount factor between any two consecutive periods; so the discount factor between the current period and the next period is $\beta_0\beta$.[85] The concept of hyperbolic discounting has a long heritage, and the tractable case of quasi-hyperbolic was popularised (not introduced) in Laibson (1997). Let's introduce the idea using a four period model. I first will describe quasi-hyperbolic here in a deterministic setting (without uncertainty), and then switch to a stochastic setting when switching to recursive notation

In a four period model with standard exponential discounting

$$u(c_1) + \beta u(c_2) + \beta^2 u(c_3) + \beta^3 u(c_4) \tag{3}$$

Notice how β is used as the discount rate between periods one-and-two as well as between periods two-and-three and three-and-four. In contrast quasi-hyperbolic discounting uses $\beta_0\beta$ between periods one-and-two, but then uses β between periods two-and-three and three-and-four. Thus our four period model with quasi-hyperbolic discounting becomes

$$u(c_1) + \beta_0\beta u(c_2) + \beta_0\beta^2 u(c_3) + \beta_0\beta^3 u(c_4) \tag{4}$$

Because we have not fully specified the optimization problem (what is being choosen, just c_1? or also c_2, c_3, & c_4?) it is not so obvious in this formulation that there are two types of quasi-hyperbolic discounter: naive and sophisticated. A *naive* quasi-hyperbolic discounter assumes that their future self will not be impatient even though their present self is (their future self is using exponential discounting). A *sophisticated* quasi-hyperbolic discounter; one who realises that their future self will also behave impatiently by discounting in a quasi-hyperbolic manner. This will be much clearer when we now switch to recursive notation and write out the optimization problem in full.

We now express the same thing in recursive notation, but adding stochastics (z and the expectations). We have that standard exponential discounting can be expressed as the value function

$$V(a,z) = max_{c,a' \in D(a,z)} u(c) + \beta E[V(a',z')] \tag{5}$$

where a is an endogenous state, and z is an exogenous stochastic state. c and a' are constrainted to be in some

[85] We use $\beta_0\beta$ as the discount rate between this period and next period, rather than just some $\alpha \equiv \beta_0\beta$ because this notation becomes much more convenient when we want to look at, e.g., life-cycle models where there is a probability of dying; especially in terms of writing the codes. Note that papers that look at theory on Quasi-Hyperbolic discounting in simple models often just use one parameter to control this period to next period, e.g., $\alpha \equiv \beta_0\beta$.

feasible decision space $D(a,z)$. In finite time $V(a',z')$ would be next period value function and so not the same as $V(a,z)$, in infinite time they are the same.

We can express naive quasi-hyperbolic discounting in terms of the exponential discounting solution. Let $\tilde{V}$ be the value function of the naive quasi-hyperbolic discounter, then

$$\tilde{V}(a,z) = max_{c,a' \in D(a,z)} u(c) + \beta_0 \beta E[V(a',z')] \tag{6}$$

notice how the expected next-period value function is that of the behavior of the exponential discounter, who the naive quasi-hyperbolic discounter (incorrectly) believes they will act like. We call V the 'continuation value' of the naive quasi-hyperbolic discounter.

The sophisticated quasi-hyperbolic discounter understands that their future self will behave as a quasi-hyperbolic discounter. Let $\hat{V}$ be the value function of the sophisticated quasi-hyperbolic discounter, then

$$\hat{V}(a,z) = max_{c,a' \in D(a,z)} u(c) + \beta_0 \beta E[\underline{\mathrm{V}}(a',z')] \tag{7}$$

and the argmax of this same expression gives the (optimal) policies $\hat{c}$ and $\hat{a}'$. Using $\hat{c}$ and $\hat{a}'$ we get the definition of $\underline{\mathrm{V}}(a,z)$ as

$$\underline{\mathrm{V}}(a,z) = u(\hat{c}) + \beta E[\underline{\mathrm{V}}(\hat{a}',z')] \tag{8}$$

note that to compute this we will need, given next period $\underline{\mathrm{V}}$, to first get current period $\hat{V}$ and $\hat{c}$ and $\hat{a}'$ from equation (7), and then use these in equation (8) to calculate the current period $\underline{\mathrm{V}}$. We call $\underline{\mathrm{V}}$ the 'continuation value' of the naive quasi-hyperbolic discounter.[86]

Notice that standard exponential discounting is nested as the case $\beta_0 = 1$, and quasi-hyperbolic discounting involves $0 < \beta_0 < 1$. For infinite horizon models $0 < \beta < 1$, but for finite horizon models with a probability of dying it can sometimes be greater than 1.

Quasi-hyperbolic preferences are time-inconsistent preferences.

[86] One way to see is going on with the continuation value of the sophisticated quasi-hyperbolic discounter is to return our trivial example of $u(c_1)+\beta_0\beta u(c_2)+\beta_0\beta^2 u(c_3)+\beta_0\beta^3 u(c_4)$, and rewrite this as $u(c_1)+\beta_0\beta[u(c_2)+\beta u(c_3)+\beta u(c_4)]$, now we can see that we have the current problem $u(c_1)+\beta_0\beta' continuation\, value'$, and the continuation value is $u(c_2)+\beta u(c_3)+\beta u(c_4)$ which only uses β and for the sophisticated agent they know that this continuation value, their $underbar V$, is based on the actions that an impatient agent will take (the 'hat' policies).

B.3 Temptation and Self-Control: Gul-Pesendorfer preferences

Under standard preferences, more options is always better. Temptation is the idea that having more options can make you worse off, and self-control captures that you can resist that temptation at a cost. Let A be a set of options, and B be another set of options, and assume A is preffered to B; standard preferences are that $A \cup B$ is preferred to both A and B, as more options is always better. The key to Gul-Pesendorfer preferences is the concept of *set betweenness*: A is preferred B, but the union is now 'between' these, that is A is preferred to $A \cup B$ is preferred to B, so adding the option of B to A makes the agent worse off.

First, let's revisit standard preferences. We can think of the standard utility of a set of options as $U(x) = \max_{x \in A} u(x)$. This view of preferences over a set of options is how we need to think to introduce Gul-Pesendorfer preferences.

Before introducing Gul-Pesendorfer preferences we need a two period setting so that we will have distinct concepts of commitment and self-control. In period 1, we think of the agent as choosing which set of options will occur in period 2. So in period one we are choosing between, e.g., A, B, and $A \cup B$. Then in period 2 we find ourselves in one of these sets, e.g., $A \cup B$, and have to choose an option from that set, e.g. choose some $x \in A \cup B$; let y be the most tempting element in $A \cup B$. Then choosing A over $A \cup B$ will be understood as commitment. Whereas, in period 2 if we are choosing a option out of $A \cup B$ then still selecting an element of x other than the most-tempting element y, will be understood as self-control. (Note: Nothing about self-control relies on this being choices from the set $A \cup B$, it could be A or B.) So commitment can be though of as avoiding temptation, while self-control is resisting temptation. Gul-Pesendorfer offer the example of whether to eat a Burger or a Sandwich for lunch (we assume the Burger is the temptation). Commitment is choosing to go for lunch at a Restaurant that only serves Sandwiches, but no Burgers: we avoid temptation. Self-Control is when we order a Sandwich for lunch in a restaurant that has both Sandwiches and Burgers on the menu: we resist temptation.[87]

So we will use a two-period setting: the first-period is a decision over sets, in the second period we choose an option from the set selected in period one. Gul-Pesendorfer preferences in the first period rank sets according to,

$$U(A) = \max_{x \in A} u(x) + w(x) - \max_{y \in A} v(y)$$

where both u and w are standard von Neumann-Morgenstern utility functions. Notice that what is being ranked by these preferences is a set A, not an element of the set. u is the *commitment ranking* and w is the *temptation ranking*. $\max_{y \in A} w(y) - w(x)$ is interpreted as the (utility) cost of temptation (and is always postitive). Choosing a lottery to maximize $u + w$ represents the optimal compromise between the utility that could have been acheived under commitment and the cost of self-control. Notice also that only the most tempting element of B matters for the (dis)utility of temptation, not anything else in B.

Gul-Pesendorfer as described so far is just preferences at period 1, but can be naturally extended to choice

[87] I don't personally find Burgers very tempting, but whatever, it is Gul and Pesendorfers example.

behaviour in the second period, when finding yourself in e.g. set A, of choosing the element in A that maximizes $u + w$; $U^*(x; A) = \max_{x \in A} u(x) + w(x)$.

The amount of commitment is determined by the 'distance' between u and w: u_1 and w_1 display greater commitment that u_2 and w_2 if the indifference curves for u_2 and w_2 are a convex combination of the indifference curves for u_1 and w_1. The amount of self-control reflects the amount of temptation that can be resisted and is determined by the 'distance' between $u + w$ and w: u_1 and w_1 display greater self-control than u_2 and w_2 if the indifference curves for $u_2 + w_2$ and w_2 are a convex combination of the indifference curves for $u_1 + w_1$ and w_1. To say the same thing in a different way, recall that in period 2 the agent maximizes $u + w$, so if $u + w$ is very different from the temptation ranking w then the agent frequently exercises self-control.

We can now extend Gul-Pesendorfer preferences to finite-horizon problems. We first do this with general notation based around the concepts of utility depending on the whole set being chosen. We will then refine it to something that we can solve. We start by defining a value function (on a different space to usual, here it is on the space of sets),

$$V(B_j, j) = \max_{d \in B_j} [u(d) + w(d) + V(B_{j+1}(d))] - \max_{\hat{d} \in B_j} w(\hat{d})$$

where d is an action, B_j is the set of possible actions this period. The action d also helps determine next period set of possible actions B_{j+1}. Note that when d determines next period state, this appears implicitly as part of determining B_{j+1}.

We can interpret $u(d) + V(B_{j+1}(d))$ as the commitment utility. $[\max_{\hat{d} \in B_j} w(\hat{d})] - w(d)$ is the (utility) cost of temptation.

We now need to rewrite this in a form that looks more like our standard approach, so that we will be able to compute the solution. This is,

$$V(a, j) = \max_{d, a' \in D(a,j)} [u(c) + w(c) + V(a', j+1)] - \max_{\hat{d}, \hat{a}' \in D(a,j)} w(c)$$
$$\text{s.t. a constraint that gives } c \text{ as a function of } d \text{ and } a'$$

notice that this now looks like our standard life-cycle problem, but with two differences: (i) the period return is now $u(c) + w(c)$, and (ii) we need to handle the $\max_{\hat{d}, \hat{a}' \in D(a,j)} w(c)$ term.

We rewrite this once more into the way the toolkit thinks about this,

$$V(a, j) = \max_{d, a' \in D(a,j)} [F_u(d, a', a) + F_w(d, a', a) + V(a', j+1)] - \max_{\hat{d}, \hat{a}' \in D(a,j)} F_w(d, a', a)$$

To implement this, we define the temptation function $F_w(d, a', a, z)$, everything else we need to know about the problem is standard. Note that $D(a, j)$ is already implicitly defined in the way the toolkit operates as any combination of (d, a') for which the temptation function is finite valued (returning -Inf is saying that $(d, a') \notin$

$D(a,z)$). When setting this up we (re)define the return function to be $F_u(d,a',a,z)$.

The extension of this to problems with uncertainty (z and e variables) is trivial,

$$V(a,z,e,j) = \max_{d,a' \in D(a,z,e,j)} [F_u(d,a',a,z,e) + F_w(d,a',a,z,e) + E[V(a',z',e',j+1)|z] - \max_{\hat{d},\hat{a}' \in D(a,z,e,j)} F_w(d,a',a,z,e)$$

B.4 Ambiguity Aversion

We can think of two types of uncertainty. *Risk*, the 'known unknowns', represents the standard approach where every possible outcome is assigned a specific probability. An alternative is *ambiguity*, the 'unknown unknowns' captures they idea that we might not be able to assign probabilities to all outcomes. The standard way to model ambiguity is known as multiple priors, and agents act based on the 'worst' prior. We still conceive of a true stochastic process for the shocks, as this is needed for simulating the model.

Let's start by defining risk, and then we will descibe ambiguity. We want to think about a possible state, ω that occurs next period, and the set of all possible states is Ω. Under standard uncertainty modeled as risk, we have some probabilities P assigned to all $\omega \in \Omega$. In a two-period setting the agent aims to maximize,

$$\max_{c_1,c_2} u(c_1) + \beta E^P[u(c_2)]$$

where E^P indicates that the expectations are taken with respect to the (subjective) belief P, which we can think of as a prior belief.

For ambiguity we instead consider as set $\mathcal{P}$ of probabilities, so the agent has 'multiple priors' about the future. We then assume that the agent acts to maximize the worst possible future under their various priors, that is they maximize,

$$\max_{c_1,c_2} u(c_1) + \beta \min_{\hat{P}\in\mathcal{P}} E^{\hat{P}}[u(c_2)]$$

So the agent is evaluating the future under each of their priors (each $P \in \mathcal{P}$), and then will choose actions that maximize the worst outcome across their priors.[88] It is worth observing that the prior used to evaluate the future (the argmin) may differ with the value of c_2.

The key trick here is taking the minimum over the expection evaluated under multiple priors. We can easily implement this in a standard finite-horizon value function problem as,

$$V(a,z,j) = \max_{d,a'} F(d,a',a,z) + \min_{\pi(z)\in\mathcal{P}} E^{\pi(z)}[V(a',z',j+1)]$$

as in this setup our 'prior belief' about the future is simply our definition of the markov process z. Hence, alongside the 'true' z, we simply define a set of 'multiple priors' in the form of alternative markov processes.

So in the code we first say that we want to use ambiguity aversion by setting *vfoptions.exoticprefererences='AmbiguityAversion'*. We then define the true markov process for z in the standard manner, which will involve both a grid and a transition matrix. It is imposed by the code that the grid for the mul-

[88] Other implementations known as 'smooth ambiguity' which also use multiple priors, but do something less extreme that taking the minimum across the multiple priors, do exist but we will not discuss them here.

tiple priors is just the same as the grid for the true process (so all of them use the same grid[89]). We set the number of the multiple priors as *vfoptions.n_ambiguity*. To set up the multiple priors is about creating numerous different version of the transition matrix, all of which get stored in *vfoptions.ambiguity_pi_z* (or *vfoptions.ambiguity_pi_z_J*), the last dimension of which indexes the multiple priors. That is it, all the trick to using ambiguity aversion is just about setting up an appropriate choice for *vfoptions.ambiguity_pi_z_J*. If you want to use i.i.d. shocks these can be set up with *vfoptions.ambiguity_pi_e* (or *vfoptions.ambiguity_pi_e_J*).

The codes are essentially going to compute $\min_{\pi(z)\in\mathcal{P}} E^{\pi(z)}[V(a',z',j+1)]$ by first using each of the multiple priors in *vfoptions.ambiguity_pi_z_J* to evalute each of the $E^{\pi(z)}[V(a',z',j+1)]$, and then take the minimum across these (in the dimension of the priors).

Notice that the 'true' z is not actually used in the value function problem. The codes still require you to define it and pass it as an input, but it does not get used (you are required to pass it as an input because the value function iteration commands expect the process on z as a standard input). The 'true' z is for simulations (simulations do not depend on the multiple priors, except via their influence on the policy function). Whether or not you include the true z as one of the multiple priors is entirely optional.

You can model a mixture of shocks as risk and shocks as ambiguity. The key is just to make it so that the multiple priors are different for the ambiguous shocks, but that all the multiple priors are identical for the risk shock.[90]

This approach to modelling ambiguity aversion was introduced by Gilboa and Schmeidler (1989), and a review is provided by Ilut and Schneider (2023).

[89] The underlying theory imposes something in a similar vein about the supports of the stochastic processes.

[90] If you take the minimum across a bunch of identical priors you will of course just get that prior. This is taking advantage of the observation that ambiguity averion nests risk aversion as the case where the set of muliple priors has only a single element. Ambiguity aversion also nests maximin preferences as the case where the set of multiple priors contains each of the P_ω which assign a probability of one to event ω.

C GMM Estimation of Life-Cycle Models

GMM estimation involves choosing/estimating the model parameters to get the 'model moments' to be close to the 'data moments'; e.g., we might estimate the parameters for an exogenous process on earnings so that the age-conditional mean earnings in the model look like those in the data. This appendix gives a formal definition of GMM estimation. It then explains things like efficient GMM, how standard deviations of the estimated parameters are calculated, local identification, and how 'sensitivity' is calculated. VFI Toolkit estimates GMM as described below in equation (10).

We begin with the formal definition of the Generalized Method of Moments, we will consider the case of *i.i.d.* data. GMM is based on a moment condition,

$$E[M(X,\theta)] = 0$$

where M is a vector-valued function, X is a random vector, and θ is the vector of parameters we want to estimate. Because we will have more moment conditions than parameters (overidentification) we cannot get all the moment conditions to equal zero and so we instead aim to minimize the weighted square of the moment conditions,

$$\theta^* = \underset{\theta\in\Theta}{\arg\min} \quad E[M(X,\theta)]' W E[M(X,\theta)] \tag{9}$$

where W is a weighting matrix. Of course to implement the estimator we have to replace the population moment conditions with the sample moment conditions, so our GMM estimator for parameter vector θ is given by,

$$\hat{\theta} = \underset{\theta\in\Theta}{\arg\min} \quad M_N(\{x_i\}_{i=1}^N,\theta)' W M_N(\{x_i\}_{i=1}^N,\theta)$$

where $M_N(\{x_i\}_{i=1}^N,\theta) = \frac{1}{N}\sum_{i=1}^N m(x_i,\theta)$ are the (vector-valued) sample moment conditions.

For a life-cycle model we are typically interested in seperable moment conditions.[91,92] In this case we can rewrite the GMM estimator as,

$$\hat{\theta} = \underset{\theta\in\Theta}{\arg\min} \quad (M^d - M^m(\theta))' W (M^d - M^m(\theta))' \tag{10}$$

where $M^d \equiv M^d(\{x_i\}_{i=1}^N) = \frac{1}{N}\sum_{i=1}^N m_d(x_i)$ are data moments, and $M^m(\theta)$ are model moments. This formulation covers the set up in most application of GMM to life-cycle models; e.g., we might have that $M^d(\{x_i\}_{i=1}^N)$ are the age-conditional mean earnings moments in the data, while $M^m(\theta)$ are the age-conditional mean earnings moments

[91] See github.com/robertdkirkby/GMMSeparableMoments for some examples of GMM estimation of simple models, which illustrate which kinds of moments conditions are seperable, and which are not.

[92] We separate the moment condition $M(X,\theta) = M^d - M^m(\theta)$ into two parts, the first part depends only on data and the second part only on the model. Note that the 'moment' of GMM is $M(X,\theta)$, but we will talk about moments of the data M^d and moments of the model $M^m(\theta)$. The confusing language in which the estimation moment is the difference between a data moment and a model moment is inherit in the way the language around GMM works.

in the model. Separable refers to the fact that our moment conditions (which depend on data and parameters) can be seperated into two seperate terms, where the first term depends (only) on the data and the second term (only) on the parameters.

Under standard assumptions, GMM is consistent and asymptotically normal, so

$$\sqrt{N}(\hat{\theta} - \theta_0) \to N(0, \Sigma)$$

where $\Sigma = (J'WJ)^{-1}J'W\Omega WJ(J'WJ)^{-1}$, where W is the weighting matrix on the moments, J is the Jacobian matrix of derivatives of model moments with respect to θ ($J \equiv \frac{\partial M^m}{\partial \theta}\big|_{\theta=\theta_0}$), and Ω is the covariance matrix of the data moments.[93,94]

Σ can be calculated by the above formula once we have J (as both W and Ω are required as inputs). J is the Jacobian matrix of the derivatives of the model moments with respect to θ, and we can compute this by finite-differences: we compute $J = \frac{\partial M^m(\theta)}{\partial \theta}|_{\theta=\theta_0}$ as $\frac{M^m(\theta_0(1+\epsilon)) - M^m(\theta_0)}{\epsilon}$.[95]

To perform the estimation using VFI Toolkit, in addition to just setting up the life-cycle model you need to declare: *EstimParamNames*, the names of the parameters you want to estimate (of θ; their initial values will be taken from the parameter structure); *TargetMoments* which is the names and values of the data moments M^d (the names of $M^m(\theta)$ are implicit in these); the weighting matrix W; and the variance-covariance matrix of the the data moments Ω.

The codes estimate the GMM by solving (10). The three main outputs are *EstimParams* which is the estimated $\hat{\theta}$, *EstimParamsConfInts* which reports the 90% confidience intervals for the estimated parameter vector, and *estsummary* which is a structure containing all sort of goodies :)

Among the things VFI Toolkit calculates and includes in *estsummary*, are J and Σ (as well as storing copies of W and Ω, so you know which ones were used in this estimation). *estsummary.objectivefnval* reports the value of $(M^d - M^m(\hat{\theta}))'W(M^d - M^m(\hat{\theta}))'$. *estsummary.EstimParamsStdDev* contains the standard deviations of the estimated parameters (the square-root of the diagonal elements of Σ). *estsummary.confindenceintervals* contains the 68, 80, 85, 90, 95, 98 and 99 percent confidence intervals (calculated from the standard deviations, as asymptotic distribution is normal).

C.1 Choosing the Weighting Matrix

GMM estimation can be performed with any positive semi-definite weighting matrix W. One popular choice is 'robust' GMM using either $W = \mathbb{I}$ (the identity matrix) or $W = diag(\Omega)^{-1}$ (zeros on the off-diagonals, and the

[93] In principle, J is the Jacobian matrix of the moment conditions for GMM, but as long as the moments are seperable, since the data moments are independent of θ, this simplifies to just the Jacobian of the model moments.

[94] In the just-identified case, where the number of moment conditions equals the number of parameters to be estimated, we simply get $\Sigma = (J'\Omega^{-1}J)^{-1}$

[95] We write *times 1 plus epsilon*, instead of *plus epsilon*, as we do the former internally to ensure the size of the 'plus epislon' is reasonable given the size of theta.

inverse of the variance of each moment on the diagonals). Note that if we change a moment measured in dollars (say mean earnings) to being measured in thousands of dollars, then this scale will matter and change the estimates if we use $W = \mathbb{I}$, but will have no effect if we use $W = diag(\Omega)^{-1}$, as the later has the desirable property of being scale independent.

C.1.1 Efficient GMM

Σ is minimized by choosing $W = \Omega^{-1}$, in which case it simplies to $\Sigma = (J'WJ)^{-1}$. This is the smallest possible Σ, so it is minimizing the standard deviations/confidence intervals for the estimated parameter vector.[96]

Note, often 'two-iteration' efficient GMM is needed for implementation, but that is not the case here because we have separable moment conditions.

C.1.2 W might cause Bias

If W is correlated with M^d, then our GMM estimator will be biased. This can be a problem in small samples.

For example, imagine we our target moments M^d include first and second moments (say, mean and variance of earnings); so we might have both $\sum_{i=1}^{N} x_i$ and $\sum_{i=1}^{N}(x_i - \bar{x})^2$ in M^d. If we use efficient GMM with $W = \Omega^{-1}$, the element which coincides with the target moment of mean earnings, will be (the inverse of) $\sum_{i=1}^{N} x_i^2$, which looks a lot like the variance moment in M^d, and so it is likely that W is correlated with M^d.

So sometimes efficient GMM will be biased in small samples; and same for some other choices of W. More generally, any time we include first and second moments among the targets we need to be careful about potentially having bias from our choice of W. In practice, a popular thing to do is just perform both robust and efficient GMM, and report both.

C.2 Parameter Identification and Sensitivity

Since the whole point of GMM is to estimate the value of the parameter vector θ we are obviously going to be interested whether the result is meaningful, and if it is fragile? Identification and Sensitivity provide formal answers to these questions.

C.3 Identification

Global identification asks whether θ^* is the unique *argmin* of the GMM objective function (9). Local identification asks whether θ^* is the unique *local-argmin*. Global identification is difficult to assess (conceptually fine, but computationally very difficult). Local identification can be evaluated by looking at the Jacobian matrix of the derivatives of the moment conditions with respect to the parameter vector, evaluated at the true parameter

[96] If you set *estimoptions.efficientW* the toolkit will use this formula, although the only difference will anyway be numerical error.

value. Because we have seperable moments this simplifies to J, which is the Jacobian matrix of the derivatives of the *model moments* with respect to the parameter vector, evaluated at the true parameter value. And we can evaluate this by computing the Jacobian matrix of the derivatives of the model moments with respect to the parameter vector, evaluated at the *estimated parameter value*, where the derivatives. Once we compute J, which was explained above, we can then simply check if $rank(J)$ is full-rank (so $\geq k$, the number of parameters being estimated), and if so then the estimate is locally-identified. VFI Toolkit does this automatically and reports it in *estsummary.localidentification*.

C.4 Sensitivity to Estimation Moments

We might be interested in how sensitive θ is to the various moments that we targetted to estimate the model. We can measure this as, roughly, the change in θ if we change the values of the moments. This is formalized by the sensitivity matrix,

$$SensitivityMatrix = -(J'WJ)^{-1}(J'W)$$

where the rows index the parameters and the columns index the moments. VFI Toolkit will compute this automatically, and reports the result in *estsummary.sensitivitymatrix*.

Like with identification, global sensitivity and local sensitivity are two different concepts, and the sensitivity matrix reported by VFI Toolkit, which was developed by Andrews, Gentzkow, and Shapiro (2017) is a measure of local sensitivity.

C.5 Sensitivity to Pre-Calibrated Parameters

Say our model has parameters θ_1 which were calibrated, and θ_2 which were GMM estimated (after the calibration was completed). We might be intereted in how sensitive θ_2 is to the calibration of θ_1. We can calculate the sensitivity matrix

$$SensitivityToCalibrationMatrix = -(J'WJ)^{-1}(J'W)J_{Calib}$$

where the rows index the estimated parameters and the columns index the pre-calibrated parameters. J_{Calib} is the Jacobian of the derivatives of the moments with respect to the pre-calibrated parameters. VFI Toolkit will do this if you set *estimoptions.CalibParamNames* to contain the names of (some) pre-calibrated parameters, and reports the results in *estsummary.sensitivitytocalibrationmatrix*. This is a local, not global, measure of sensitivity, and was developed by Jorgensen (2023).